Titre: Title:	Modèles et algorithmes de résolution du problème de chargement des aéronefs
Auteur: Author:	Léandre Ratsirahonana
Date:	2003
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ratsirahonana, L. (2003). Modèles et algorithmes de résolution du problème de chargement des aéronefs [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7266/

UNIVERSITÉ DE MONTRÉAL

MODÈLES ET ALGORITHMES DE RÉSOLUTION DU PROBLÈME DE CHARGEMENT DES AÉRONEFS

LÉANDRE RATSIRAHONANA
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR (Ph.D.)
(MATHÉMATIQUES DE L'INGÉNIEUR)
AOÛT 2003

Your file Votre référence
ISBN: 0-612-89233-6
Our file Notre référence
ISBN: 0-612-89233-6

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

MODÈLES ET ALGORITHMES DE RÉSOLUTION DU PROBLÈME DE CHARGEMENT DES AÉRONEFS

présentée par : RATSIRAHONANA Léandre
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. DESAULNIERS Guy, Ph.D., président
M. SAVARD Gilles, Ph.D., membre et directeur de recherche
M. CHAUNY Fabien, Ph.D., membre et codirecteur de recherche
M. GAMACHE Michel, Ph.D., membre
M. BENCHAKROUN Abdelhamid, Ph.D., membre externe

À Michèle, ma bien-aimée...

REMERCIEMENTS

Je tiens à exprimer ma gratitude envers mon directeur de recherche, le professeur Gilles Savard, pour sa confiance, sa générosité, sa disponibilité ainsi que l'aide morale et matérielle qu'il m'a accordée tout au long de cette thèse de doctorat.

J'exprime ma reconnaissance au professeur Fabien Chauny, mon codirecteur de recherche, pour ses nombreux et précieux conseils, pour tout le temps qu'il a consacré et pour l'intérêt qu'il a porté à cette recherche.

Je voudrais remercier le professeur Guy Desaulniers qui a accepté de présider le jury de cette thèse, le professeur Michel Gamache, membre du jury et le professeur Abdelhamid Benchakroun, le membre externe du jury.

Un remerciement bien particulier à Daniel Villeneuve pour l'aide et les discussions lors des implantations des modules. Je voudrais aussi remercier les amis brésiliens, tous mes collègues du GERAD et du département de mathématiques et de génie industriel qui, de près ou de loin, ont collaboré à ce projet.

Je remercie ma famille pour ses encouragements tout au long de mes études. Je dédie une pensée particulière à Michèle qui a partagé mes espoirs et mes doutes dans la réalisation de cette thèse.

RÉSUMÉ

Le projet de recherche qui suit concerne le développement de modèles de planification et d'optimisation de chargements des avions-cargos pour satisfaire à la demande avec une flotte hétérogène d'avions-cargos tels les *Hercules CC130*, appareils utilisés par *les Forces Armées Canadiennes*. Nous proposons dans cette dissertation un modèle beaucoup plus réaliste que ceux qu'on retrouve dans la littérature. Ce modèle tient compte des positions de placement des items dans l'avion, de la diversité de sa cargaison, des différents lieux de collectes des items. Certaines extensions, présentées comme des cas spéciaux, entre autres, le transport de passagers, les considérations de priorités d'envoi des items, le cas où les items ont une densité de masse non uniforme et les contraintes de préséance peuvent être traitées aussi par ce modèle. Une des principales caractéristiques de ce modèle est qu'il tient compte des contraintes sur la position du centre de gravité de l'avion chargé. Cette contrainte a un impact direct sur la sécurité du vol ainsi que sur des économies de carburant. Le problème est traité en deux étapes : la résolution du problème avec un seul point de collecte et une seule destination de livraison et la résolution du problème avec plusieurs points de collecte et une seule destination de livraison.

Le problème a été modélisé sous forme d'un problème de flot multi-commodités avec des contraintes supplémentaires. Comme certaines contraintes sont non convexes, le modèle est résolu par un algorithme de séparation et évaluation progressive (*branch-and-bound*) où les bornes inférieures sont calculées par génération de colonnes. Les colonnes correspondent à des plans de chargement qui respectent les contraintes opérationnelles. Nous avons eu recours au logiciel *GENCOL* pour résoudre le problème. Cependant de nouveaux modules ont été développés au niveau de ce logiciel pour pouvoir l'adapter à notre problème.

Vu la taille d'un problème réel d'environ cinq villes de collecte, quelques centaines d'items à transporter regroupés en quelques dizaines de types différents, nous avons construit un réseau de chargement sur lequel on développe les équations du modèle mathématique. Nous avons réalisé une procédure de réduction de la taille en termes d'arcs ou de nœuds du réseau. L'élimination de certaines symétries du problème a été aussi étudiée. Ces prétraitements permettent de réduire le saut d'intégrité dans le modèle.

Les résultats des tests effectués, entre autres, sur le problème réel de Ng (1992) indiquent sa résolution à l'optimalité en moins de 6 minutes et une réduction de $2 millions par rapport à celle de Ng et de $3 millions par rapport à la solution proposée par les maîtres de chargement (*loadmasters*). D'autres problèmes générés aléatoirement ont été aussi résolus à l'optimalité dans des temps raisonnables.

Pour améliorer le temps de résolution et réduire la mémoire requise pour chaque instance étudiée, nous avons agrégé des contraintes liant les items de même type. Le modèle reste toujours celui d'un réseau multi-commodités, mais la formulation utilise des variables réelles et en nombres entiers, pas seulement binaires. La décomposition de Dantzig-Wolfe est appliquée à cette nouvelle formulation pour fournir la borne inférieure de la relaxation linéaire. Le problème maître comporte une seule contrainte de couverture pour les items de même type. La structure du sous-problème reste inchangée. Les résultats obtenus pour l'ensemble des tests montrent que tous les problèmes ont été résolus à l'optimalité, mais non garantis. Les temps de calcul sont de 140 fois plus rapides en moyenne par rapport à l'approche non agrégée, et cette approche nécessite à peu près 15 fois moins de mémoire.

Un modèle hybride a été développé. Le problème maître est modifié et ne comporte qu'un nombre minimum de contraintes avec second membre fixé à 1, et éventuellement, une contrainte avec second membre plus grand ou égal à 1. Une méthode de

branchement est proposée. La règle s'inspire de celle de Desrochers et Soumis (1989) et elle élimine des problèmes de symétrie, améliorant ainsi l'efficacité des décisions de branchement. Nous avons implanté, testé et comparé notre méthode avec celle de l'approche non agrégée sur les mêmes données. Les temps de calcul sont de l'ordre de 4 fois plus rapide en moyenne par rapport à l'approche non agrégée et les gains en termes de mémoire consommée sont de l'ordre de 80%. Le mérite de l'algorithme est qu'il s'adapte facilement, sans modification, pour les problèmes formulés sous forme partitionnement d'ensembles où le second membre est un nombre entier plus grand ou égal à 1.

L'étude du transport des items situés en des points géographiques différents vers un point particulier est abordée par la suite. La formulation unifiée sert encore de modèle et des extensions ont été réalisées pour tenir compte des villes et des distances entre elles. Deux possibilités de chargement sont proposées dans le modèle. Dans le premier cas, les items sont chargés selon l'ordre de visite de leur ville de collecte et aucun arrangement des positions n'est possible entre les items chargés dans les villes intermédiaires. Dans le second, des réarrangements sont permis. Ce dernier est moins restrictif que le premier et plus de plans réalisables sont considérés. Par contre, il implique des procédures d'interdiction de plans. Chaque formulation est de nouveau résolue par la méthode de génération de colonnes. Le problème maître comporte une ou plusieurs contraintes de couverture pour les items de même type. La structure du sous-problème reste inchangée : celle d'un problème de plus court chemin avec contraintes de ressource.

Le jeu de données réelles de Ng et les instances générées aléatoirement et représentatives de problèmes réels sont réutilisés pour effectuer les tests numériques. Des études sur l'impact du coût élevé associé à la visite de deux villes successives de collecte ou sur l'impact de la géographie des villes de collecte sont effectuées. Enfin, une étude du comportement des algorithmes proposés, lorsque les items à transporter varient et

lorsque le nombre de villes augmente, termine les séries de tests. Les résultats obtenus démontrent qu'il a été possible de résoudre de façon satisfaisante le modèle avec la méthode de génération de colonnes. Pour chaque instance, une bonne solution a été trouvée dans un temps relativement court et sans trop utiliser de mémoire.

ABSTRACT

This dissertation deals with the development of aircraft loading planning and optimization models in order to satisfy demands with an heterogeneous aircraft fleet such as the *Hercules CC130* used by *Canadian Air Force*. We propose the use of a more realistic model than those found in the literature. Our proposed model accounts for the position of items in the aircraft, the diversity of each load, the different pick-up locations. Further extensions are presented as special cases and can be treated by this model, such as the transportation of passengers, the consideration of shipping priority, the case of items with non uniform mass density and the constraints of precedence. We consider the center of gravity (CG) of each item and we study the CG of the load/aircraft as well, in other words, the balancing of the aircraft. The problem is solved in two steps: the resolution of the problem with only one pick-up location and one delivery and the problem with several pick-ups and one delivery.

The problem has been modeled as a multi-commodity network flow problem with additional constraints. Because of the non-convexity of the solution domain, the model is solved within a *branch-and-bound* framework. The computation of lower bounds are made by column generation where each column corresponds to a path followed by some commodity on the network. Each path is a loading plan that obeys the operational constraints of loading. We have used the software *GENCOL* to solve the problem. However, new modules were developed to adapt it on our problem.

Actual size of a problem involves 5 pick-up cities, about 20 types of items involving many hundred items to be carried. We use a network of loading in which the mathematical model equations lie. Some procedures are realized to reduce the size of the network in terms of arc and node. The elimination of some symmetry problems is

also studied. These preliminary treatments allow to reduce the integrity gap for the model.

To guarantee the flight security and to save on fuel, the CG of the aircraft must fall within a predefined range. This particularity makes the modeling more difficult and presents a risk for suboptimal solution if all potential plans are not considered. We have studied this balancing problem along the longitudinal axis of the aircraft. Two resource variables control respectively the lower and upper limit constraints for the CG in the resolution of the shortest path in each sub-problem. To achieve that any potential plan would not lost during the process, each generated plan has conserved and process of detecting dominants labels is suggested.

The test results show that the real problem of Ng (1992) is solved at optimality within less than 6 minutes and $2 millions reduction compared with of Ng and with $3 millions compared with the proposed solution of the *loadmasters*. Other problems generated randomly were also solved at optimality.

To improve the time of resolution and to reduce the request memory for the resolution of each instance, we have aggregated constraints related to some items. The model remains the same as a multi-commodity network flow problem, but the formulation uses real and integer variables not restricted to be binary. The Dantzig-Wolfe decomposition is applied to this formulation to provide the lower bound of the linear relaxation. The master problem is composed of only one constraint of demand for items of same type making the right-hand side greater or equal to 1. The structure of the sub-problem remains unchanged. The criteria of Vanderbeck specialized for the cutting stock problem have been implemented and tested on the same data. The test results show that all problems were solved at optimality without any guaranty. The calculation time are 140 times faster in average compared with the non-aggregated approach and a saving of about 15 times less in used memory is made.

In the optic of an optimal approach, the master problem is modified and is only composed of a minimum number of constraints with right-hand side fixed to 1 and eventually one constraint with second hand side greater or equal to 1. One rule of branching is proposed. It is a generalization of the Desrochers et Soumis (1989) and it eliminates some symmetry problems, improving the effectiveness of branching decisions. We have implemented, tested and compared our method with the non-aggregated approach for the same data. The calculation times are about 4 times faster in average and the gains in term of memory used are about 80%. The merit of the algorithm is that it is easily adaptable for the problem formulated as general partitioning problems.

Transportation of items located in various geographical points is studied next. The unified formulation model is used and extensions have been made to account for cities and the cost of transport between them. Two possibilities of loading are proposed in the model. In the first case, the items are loaded according to the order of city visits and no position arrangement is possible in intermediate city. For the second case, arrangements are allowed. This later is less restricted than the former and more realizable plans were considered. On the other way, it implies forbidden plans. Each formulation is solved by a column generation algorithm. The master problem is composed of one or many demand constraints for items of the same type. The sub-problem structure remains the same: a shortest path problem with resource constraints. Since the size of each instance is large, and to find a good solution, we propose exact algorithm and heuristic to solve them.

The real data set of Ng and generated randomly problems representative of real problems were used to carry out the numerical tests. Studies on the impact of high cost associated with the visit of two successive pick-up cities or on the impact of geographical pick-up cities were carried out. Finally, a study of comportment of the proposed algorithm when the transported items vary and when the number of city

increases completes the series of tests. Results show that it was possible to solve with satisfactorily the model using the column generation approach. For each instance, a good solution has been found within a relatively short time and without using too much CPU memory.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

INTRODUCTION

Depuis quelques décennies, la diversité, la quantité toujours croissante de la cargaison à transporter et la hausse des coûts du fuel ont contribué à l'intérêt de développer un système de planification du chargement permettant de réduire les coûts des opérations. Ainsi, dans les organisations aériennes civiles, une croissance sur le nombre et la taille de la cargaison aéroportée peut créer des pressions pour le personnel des services affectés à ces tâches, entraînant par là-même un besoin continu d'améliorer la méthodologie et la technologie des moyens logistiques dans les aéroports (Larsen et Mikkelsen, 1980).

L'objectif des organisations aériennes aussi bien civiles que militaires est d'utiliser efficacement l'espace disponible de l'avion tout en assurant correctement le transport des cargaisons. Les organisations militaires doivent cependant tenir compte d'autres considérations. En temps de guerre, par exemple, leurs principales préoccupations sont de veiller à la rapidité et aux facteurs de sécurité liés à l'exécution d'une mission, celle-ci étant le déploiement d'un nombre donné de troupes et de quantités de matériels sur un ou plusieurs avions affectés à cet effet. En temps de paix, la minimisation du nombre d'avions requis pour accomplir une mission est plus significative en termes de réduction de coût. Ainsi, épargner un avion-cargo *Hercules CC130*, un appareil utilisé par les Forces Armées Canadiennes, partant de Trenton, Ontario vers la Norvège permet d'économiser plus de $100 000 (Ng, 1992) et de sauver un temps précieux. Toujours, dans le but de minimiser des coûts, déterminer les routes optimales constitue une problématique importante. En effet, ces routes s'étalent sur de grandes distances avec de multiples étapes. En outre, les facteurs tels le prix de l'essence ou le prix d'atterrissage qui varient d'un pays à l'autre ou d'une base militaire à une autre, l'entretien des appareils qui ne peut être effectué que sur certaines

bases, doivent être prises également en compte. Enfin, à distances parcourues égales, certaines routes sont préférées à d'autres car elles passent, par exemple, par une base qui offre un ravitaillement bon marché et de bonnes conditions pour l'entretien.

On sait également que la planification d'un ou de plusieurs chargements prend beaucoup de temps à cause de nombreuses contraintes opérationnelles à respecter, aussi cette tâche est-elle laissée à un maître de chargement (*loadmaster*) qui peut prendre de 30 minutes à une heure pour planifier de façon empirique un seul chargement (Cochard et Yost, 1985). Au cas où l'on changerait la liste des items à charger à la dernière minute, comme cela arrive souvent, c'est tout le processus qui est remis en question. Ainsi, des systèmes automatiques avec éditeur graphique ont vu le jour dès la fin des années 1970. L'utilisateur peut déterminer le placement des cargaisons à l'intérieur des avions, fournissant de la sorte une estimation du chargement (Anderson et Ortiz, 1978). D'autres outils assistés par ordinateur sont développés par la suite, mais ils sont considérés comme un assistant informatique plutôt qu'un générateur de plans ou une automatisation du processus de la planification des chargements.

Le projet de recherche qui suit concerne donc la réalisation d'un système de planification et d'optimisation de chargements des avions-cargos pour satisfaire à la demande avec une flotte hétérogène d'avions-cargos tels les *Hercules CC130*, appareils utilisés par *les Forces Armées Canadiennes*. Nous proposons dans cette thèse un modèle beaucoup plus réaliste que ceux que l'on retrouve dans la littérature. Ce modèle tient compte des positions de placement des items dans l'avion, de la diversité de sa cargaison, des différents lieux de collectes des items, du transport de passagers, du cas où les items ont une densité de masse non uniforme et des contraintes de préséance entre les items. Nous considérons le centre de gravité de chaque item et nous étudions aussi le centre de gravité de l'ensemble avion/cargo ou en d'autres termes le balancement de l'ensemble. A priori, minimiser le nombre d'avions et assurer le balancement de l'ensemble avion/cargo semblent contradictoires. Il faut donc trouver

un compromis, ce qui n'est pas toujours évident vu la taille des problèmes à résoudre et les multiples choix possibles de cargaisons réalisables. Le problème de balancement de l'avion et du cargo est traité sur un axe longitudinal de l'avion qui est déjà un problème assez difficile en soi. Nos hypothèses de recherche se basent alors sur des items de largeurs assez grandes de sorte qu'on ne puisse pas mettre deux items côte à côte. Ces problèmes qui sont le sujet de cette thèse sont traités en trois étapes :

1. Étude d'une stratégie optimale qui surmonte les difficultés citées plus haut pour le cas particulier d'une ville de collecte.
2. Amélioration des temps CPU de résolution et/ou de la réduction de la mémoire requise par une méthode d'agrégation des contraintes.
3. Étude du cas général où plusieurs villes de collecte sont envisagées.

Notre contribution se situe dans la modélisation du problème en la rendant plus réaliste et dans sa résolution en faisant appel à des algorithmes spécialisés.

Dans le premier chapitre, nous procédons à la description du problème, et par la suite, nous présentons la revue de la littérature ainsi que l'approche proposée et d'autres alternatives pour résoudre le problème.

Le deuxième chapitre présente un modèle de résolution du problème de chargement des avions-cargos entre un point de collecte et un point de livraison. C'est un problème de flots multi-commodités avec des contraintes portant sur les ressources. Sa résolution se fait de façon exacte au moyen d'un processus de séparation et d'évaluation progressive où les bornes inférieures sont calculées par génération de colonnes. Le problème est défini sur un graphe et la résolution d'un exemple tiré du problème de Ng (1992) illustre les différentes étapes de la construction du réseau de chargement pour lequel on établit les équations du modèle mathématique. Ces étapes constituent un prétraitement pour notre étude car elles mettent en valeur une procédure de réduction de la taille en termes d'arcs ou de nœuds du réseau ou encore l'élimination

de certaines symétries du problème. Par la suite, les équations du modèle mathématique sont posées et les variables de ressource qui modélisent les contraintes locales de chargement en les rendant plus réalistes sont explicitées. Dans la recherche d'une solution entière, des coupes sont proposées et les stratégies de branchement de Desrochers et Soumis (1989) pour trouver la borne supérieure sont utilisées. Des tests sont effectués sur un problème réel et des problèmes générés aléatoirement pour montrer la performance de l'algorithme.

Le troisième chapitre porte sur la résolution d'un modèle issu du premier et s'appuyant sur l'agrégation de certaines contraintes liant les items de mêmes caractéristiques. Le modèle admet une formulation utilisant des variables réelles et en nombres entiers. La décomposition de Dantzig-Wolfe est appliquée à la formulation pour obtenir une borne inférieure de la relaxation linéaire. Le problème maître comporte une seule contrainte de demande de chargement pour chaque type d'item. La structure du sous-problème est celle d'un problème de plus court chemin avec contraintes de ressource. Le critère de Vanderbeck (2000) est utilisé pour résoudre le modèle. Le branchement est basé sur l'identification d'un ensemble de patrons qui utilisent un arc bien précis du réseau et dont la somme des variables est fractionnaire. Le branchement peut être adapté à notre problème, mais un contre-exemple montre qu'il n'est plus optimal pour notre application. Dans l'optique d'une résolution exacte du problème, nous proposons une modification sur les contraintes de demande de chargement pour chaque item. Dans le problème maître, nous mettons un nombre minimum de contraintes avec second membre fixé à 1, et éventuellement, une contrainte avec second membre plus grand ou égal à 1. La règle de branchement s'inspire de celle de Desrochers et Soumis (1989) et elle consiste à charger deux types d'items, l'un suivi de l'autre dans un plan, ou de l'interdire. Les plans de chargement sont construits un à un. De ce fait, la sélection de la prochaine variable de branchement se fait en fonction du prolongement d'un plan de chargement déjà commencé. Le plan complet

est mis à l'écart et un problème résiduel est résolu par la suite. La procédure est réappliquée au problème résiduel courant jusqu'à l'obtention de la solution optimale. Des tests sont aussi effectués sur le problème réel et les problèmes générés aléatoirement pour montrer la performance des deux algorithmes.

L'étude du cas général où les items sont collectés sur plusieurs villes est abordée dans le quatrième chapitre. Les coûts de transport entre les villes et les contraintes opérationnelles de chargement sont pris en compte. Nous proposons deux modèles qui sont des extensions des modèles agrégés. Chaque modèle construit les plans de chargement réalisables pour tous les items et les itinéraires suivis par les avions-cargos concernés, de telle sorte que les coûts totaux de transport soient minimaux. Dans le premier modèle, les items sont chargés selon l'ordre de visite de leur ville de collecte et aucun arrangement des positions n'est possible entre les items chargés dans les villes intermédiaires. Un second modèle qui permet des réarrangements des positions entre les items déjà chargés dans les villes visitées est ensuite présenté. Ce dernier est moins restrictif que le premier et permet de considérer plus de plans réalisables, mais il implique des procédures d'interdiction de plans. Pour trouver une bonne solution entière aux problèmes tests, nous utilisons soit le critère de Vanderbeck (2000), soit la méthode développée dans le chapitre précédent. Nous proposons aussi une heuristique qui trouve en un temps relativement court une solution satisfaisante avec le nombre d'avions minimal.

Le dernier chapitre est consacré à la conclusion et nous reprenons les principales contributions de la thèse. Nous y mentionnons quelques avenues de recherche.

CHAPITRE 1 : DESCRIPTION DU PROBLÈME ET REVUE DE LA LITTÉRATURE

1.1 La description du problème

Le problème étudié consiste à transporter par avions-cargos, en partant de bases différentes vers une base de destination commune, une liste spécifique de cargaisons suivant une séquence de priorités d'envoi, tout en minimisant les coûts totaux des opérations. Ce problème, que nous appelons problème de chargement des avions-cargos, est connu en anglais sous l'appellation *Aircraft Loading Problem* ou *ALP*.

Pour le type d'avion *Hercules CC130*, le chargement de la cargaison se fait par une porte située à l'arrière, sous l'avion. Cette porte est appelée la rampe. Les items chargés sont alors placés dans un compartiment appelé le quai. Certains items peuvent être chargés sur la rampe au moment où celle-ci se relève. Les items chargés sur le quai ou sur la rampe sont soumis à certaines restrictions ou contraintes opérationnelles de chargement. Ainsi, la longueur de la cargaison chargée sur le quai ne peut pas dépasser la longueur du quai, la longueur totale de la cargaison ne peut pas dépasser celle de l'avion, et le poids total du chargement ne doit pas dépasser la charge maximale supportable par l'avion. Les items mis sur la rampe sont assujettis à certaines limites de poids, de hauteur et de longueur. Par ailleurs, une mesure du centre de gravité (CG) du cargo permet de valider le chargement. En effet, le CG de l'ensemble doit tomber dans une enveloppe prédéfinie, car un déséquilibre peut, dans certains cas, provoquer l'instabilité de l'avion. Il n'est pas impératif que le chargement soit parfaitement équilibré, car le pilote peut toujours ajuster le balancement en cours de vol. Cependant, cela peut développer une résistance à l'avancement, ce

qui a pour effet de baisser la vitesse et d'augmenter la consommation en carburant (Chapel, 1948). Pour avoir une idée de la pertinence du problème, voilà ce que disent Mongeau et Bès (2000) : "un déplacement du CG de moins de 75 cm sur le sens de la longueur d'un Airbus A340-300 fournit un gain de 4000 kg de fuel sur un vol de 1000 km".

D'autres aspects du problème doivent aussi être considérés, telle la diversité de la cargaison. Cette dernière peut comprendre la liste suivante : des véhicules à roues motorisés, des articles sur palettes, des véhicules remorques, des hélicoptères, des équipements divers et du personnel. Tous les items doivent être bien attachés une fois dans l'avion. Notons que les hélicoptères sont démontés avant d'être embarqués. En outre, les petits items sont placés à l'intérieur des gros items et les moyens items doivent être déposés sur des palettes avant d'être chargés dans l'avion. La mise sur palettes ou palettisation, un problème qui ne nous concerne pas, a déjà été étudiée par plusieurs auteurs. Nous renvoyons le lecteur aux travaux de Bischoff et Ratcliff (1995), et Gehring *et al.* (1990).

Par ailleurs, un nombre suffisant de sièges mobiles pourrait être fixé à l'avant de l'avion pour placer les passagers (personnel, militaires, etc.) avec leurs bagages respectifs. Ces sièges sont placés par rangée et de ce fait, nous pouvons considérer une ou plusieurs rangées comme un item spécial. Ainsi, nous supposons par la suite que chaque cargaison est composée de "gros items" et/ou de palettes de telle sorte que nous ne puissions mettre deux items quelconques côte à côte, et dès lors, nous ne considérons pas la largeur d'un item dès qu'il rentre dans l'avion. Cette hypothèse revient alors à étudier le balancement de l'avion suivant uniquement son axe longitudinal. Cette approche unidimensionnelle semble réaliste car selon Chapel (1948) : "il est avantageux d'équilibrer le chargement de l'avant vers l'arrière, car en général, balancer le chargement d'un côté à l'autre n'est pas très significatif, aussi seul le balancement dans le sens de la longueur de l'avion doit être pris en compte". Précisons

également que la longueur et le poids de chaque item sont connus à l'avance ainsi que la position du CG de l'item et l'information sur la possibilité pour l'item d'être placé sur la rampe ou non. Un ordre de priorité d'envoi est aussi associé à chaque item. Ces priorités sont des nombres entre 1 et 10 ; idéalement tous les items de priorité 1 doivent être transportés avant les items de priorité 2, et ainsi de suite.

Dans certaines situations, comme par exemple en temps de guerre, des items peuvent être déchargés par parachute en des endroits où il n'y a pas d'aéroport. Dans ce cas, la séquence de chargement des items à l'intérieur de l'avion selon les sites de débarquement et l'itinéraire doivent être considérés. Additionnellement, d'autres conditions d'exclusivité et/ou de préordonnancement sur les items peuvent être requises pour raison de sécurité ou de logistique. Par exemple, on placera une remorque avec le véhicule qui la tire. De même, les articles magnétiques ou dangereux sont placés dans des positions restreintes sécuritaires et enfin, l'essence et les vivres ne seront jamais affectés dans le même avion.

Ces contraintes ne sont pas exhaustives, pour une description plus générale du problème, le lecteur peut consulter Martin-Vega (1985).

Nous complétons cette description du problème par une revue de la littérature du problème de chargement.

1.2 La revue de la littérature

Les problèmes qui consistent à placer des objets de tailles données dans des contenants de capacité connue sont classiques. Les objets et les contenants sont décrits de façon très diverse. Ainsi, dans les problèmes de stockage de programmes en mémoire ou de fichiers sur un disque, ou dans les problèmes de rangement de caisses dans des

conteneurs ou sur des étagères, etc., les objets et contenants sont respectivement les programmes et la mémoire, les fichiers et le disque, les caisses et conteneurs ou étagères. Notons que les objets cités sont non fragmentables et peuvent coexister dans un même contenant (les caisses dans un conteneur par exemple). Ces problèmes sont connus sous l'appellation de problèmes de conditionnement ou *packing problems.* Lorsque les objets sont fragmentables ou représentent des produits fluides (liquides et gaz dans des réservoirs, poudres dans des silos, etc.) mais non miscibles dans un même contenant, les problèmes sont connus sous l'appellation de problèmes de chargements ou *loading problems.*

Les problèmes qui consistent à découper un ensemble d'objets dans des contenants sont des problèmes très voisins de ceux cités plus haut. Ils sont appelés problèmes de découpe ou *cutting stock problems.* On pense aux problèmes de découpes de barres d'acier ou de découpes de tôles ou de contreplaqués dans des grandes plaques-mères.

Les problèmes qui consistent à placer ou à découper un ensemble d'objets dans des contenants sont appelés collectivement des problèmes de placement ou *cutting and packing problems.* Dans les cas les plus simples, les tailles et capacités ont une seule dimension (poids d'une caisse, longueur d'un câble, etc.) et les contenants ont la même capacité. Des problèmes plus compliqués existent aussi dans la littérature, par exemple avec plusieurs types de contenants et avec objets à deux ou trois dimensions. Ces dimensions sont souvent leur repésentation géométrique (par exemple : plaques, boîtes, etc.), ou des caractéristiques physiques (poids, volume, temps, etc.). Les critères fréquemment utilisés consistent à remplir au maximum un contenant ou à placer tous les objets dans un nombre minimal de contenants, ou encore à équilibrer les charges ou à minimiser les chutes (*trim loss*) dans les problèmes de découpe. Lorsque les problèmes reviennent à maximiser la valeur de placement ou de découpage des objets dans des contenants, avec l'unique contrainte de capacité sur le contenant, ils sont appelés problèmes de sac à dos ou *knapsack problems.* Ces

problèmes apparaissent dans plusieurs secteurs industriels tels l'acier, le bois ou le papier, les vêtements, la logistique pour ne citer que ceux-là. Pour plus d'informations sur la typologie entre les problèmes de placement, le lecteur peut consulter l'article de Dyckhoff (1990). Dyckhoff *et al.* (1996) présentent une large bibliographie commentée des publications parues après 1990, avec quelques exceptions pour les articles qui ont marqué le développement de l'art concernant ces problèmes. Ils se réfèrent essentiellement au livre de Dyckhoff et Finke (1992) pour les années avant 1990. Sweeney et Paternoster (1992) proposent aussi une bibliographie très complète sur ces problèmes.

Les premières études concernant les problèmes de chargement se sont concentrées sur des problèmes de petite taille. Eilon et Christofides (1971) ont étudié des problèmes particuliers apparentés aux problèmes de sac à dos. Les applications de leur modèle concernent les problèmes de chargement de véhicules ou autres contenants, lorsque les items ne sont pas fragmentables, ou encore les problèmes de distribution lorsque les grands objets doivent être répartis en des petits lots, en respectant certaines contraintes de demande. Ils proposent deux méthodes : la première est un modèle de programmation linéaire à variables binaires et résolue par une méthode exacte de séparation et évaluation progressive ou *branch-and-bound (B&B)* et la seconde est un algorithme heuristique. Leur approche peut traiter des problèmes à plusieurs dimensions ou encore peut s'étendre au cas des contenants ayant des capacités différentes. La taille des problèmes résolus par les deux méthodes limitent leur approche. En effet, leur expérimentation consiste d'abord à tester 50 problèmes à une dimension avec plus de 50 items et 12 autres problèmes additionnels à plusieurs dimensions comportant au plus 40 items, et à comparer ensuite l'efficacité des deux méthodes. Ils concluent que, à l'exception de quelques problèmes, l'heuristique trouve la solution optimale dans des temps significativement courts par rapport à la méthode exacte.

Martin-Vega (1985) donne une description détaillée et technique de la problématique de *ALP* dans les contextes civil et militaire. Dans un premier temps, il relate les

différentes étapes et problèmes rencontrés lors du chargement, lorsque le processus se fait par essais et erreurs, et dans un deuxième temps, il revoit les travaux pertinents sur le développement d'un système informatique et enfin il présente les résultats de l'utilisation de cette approche. Il note que le point de départ théorique de nombreux travaux est celui de la généralisation du problème de sac à dos, dont l'objectif est vu comme la maximisation des objets pouvant être chargés dans un conteneur. Le problème de sac à dos étant reconnu $\mathcal{NP}$-difficile, par réduction (Garey et Johnson, 1979) de ALP au problème de sac à dos, ALP est aussi $\mathcal{NP}$-difficile dans le sens fort. Dès lors, trouver la solution optimale peut s'avérer moins rapide qu'une méthode intelligente ou heuristique mais qui ne garantit pas l'optimalité de la solution proposée. De même, il est très difficile de trouver une solution optimale lorsque la quantité de la cargaison devient trop grande et s'avère parfois impraticable en termes de temps de calcul. Ainsi, développer un chargement "optimal" n'a pas toujours été le principal objectif des planificateurs, lesquels ont préféré générer de bons plans réalisables plus rapidement.

A partir de 1980, des procédures informatiques interactives voient le jour. Dans le transport aérien civil, Larsen et Mikkelsen (1980) ont développé *CARLO* (CARgo LOading) pour générer des plans de chargement pour les vols transatlantiques des avions Boeing 747 Combi de la Scandinavian Airlines, dont le quai de chargement est divisé en 6 ou 12 compartiments. Leur objectif est double : le plan de chargement doit contenir un nombre minimal de modifications ou changements de positions des items aux aéroports intermédiaires, et le CG doit être le plus proche du milieu de l'intervalle admissible. Ils ont formulé le problème en un modèle mathématique avec des variables 0-1 et des contraintes non linéaires, sans le décrire explicitement. Par ailleurs, deux heuristiques sont incorporées dans *CARLO* pour générer un plan de chargement ; la première heuristique est basée sur un principe d'amélioration et la seconde sur un principe de construction. Le planificateur peut choisir un plan de chargement ou l'autre ou encore rejeter les deux si ceux-ci ne satisfont pas aux contraintes.

Dans ce dernier cas, la procédure est alors répétée. La version courante de CARLO est conçue pour le besoin de la compagnie aérienne. Le nombre maximum de types d'items qu'on peut traiter est de 7, le nombre d'étapes (*leg*) pour un vol est limité à 2. Il s'agit probablement de la plus sérieuse limitation en ce qui concerne cette version et une extension possible peut augmenter d'une manière exponentielle le temps de calcul. *CARLO* a été testé avec succès sur des données réelles en des temps relativement courts, entre 5 et 10 minutes comparativement à une durée d'une heure avec l'approche manuelle; le système a été couramment utilisé par la Scandinavian Airlines.

Dans le transport militaire, Huebner (1982) est le premier à proposer aux planificateurs des systèmes informatiques pouvant, d'une part, générer rapidement et avec une précision relative des chargements "quasi-optimaux", et de l'autre, améliorer l'effort logistique. Le projet a connu un succès durant un exercice militaire en Egypte. Un des concepts expérimentés pour l'étude du CG est le chargement en pyramide. Il s'agit de mettre le plus lourd item de la liste à charger au centre du quai et d'alterner les items suivants en avant ou en arrière selon la place disponible. L'auteur mentionne que l'implantation du modèle est simple.

Les idées d'Huebner sont exploitées et développées par Cochard et Yost (1985) pour un système appelé *DMES* (*Deployment Mobility Execution System*). Le modèle utilise une heuristique modifiée du problème de découpe pour générer des chargements valides, que le planificateur peut modifier à travers des traitements graphiques interactifs. L'objectif est de remplir le plus possible l'avion, tout en s'assurant de ne pas violer les contraintes tels l'enveloppe réalisable du CG, les restrictions de hauteur, le poids maximal, les restrictions sur les poids supportés par les axes ou *axle-weight*, les items incompatibles et la facilité de chargement et de déchargement. L'heuristique commence par trier les items, supposés rectangulaires, en ordre décroissant de longueur. Le plus long des items disponibles est placé à l'avant de l'avion et l'heuristique

coupe le plancher de l'avion à la fin de l'item. L'item suivant, le plus long parmi ceux susceptibles de s'insérer à côté du premier, est choisi. L'heuristique coupe encore une fois le plancher et recherche un autre item, toujours le plus long, pouvant être placé dans l'espace restant. Le procédé est répété jusqu'à ce que l'espace soit rempli ou qu'aucun item ne puisse se placer. Un premier bloc est alors défini par ce procédé et un autre sera créé tant qu'il reste des items à charger. Chaque bloc est considéré comme un item large et une fois que le plancher de l'avion est rempli, on permute ou on fait glisser des blocs à l'intérieur de l'avion pour avoir un centre de gravité acceptable. Ignizio (1991) souligne que *DMES* répond aux définitions et aux objectifs d'un système expert. Le système a été apparemment implanté avec succès lors des opérations de sauvetage à Grenada en 1983 et dans d'autres exercices en temps de paix. Les résultats rapportent une croissance de 10% sur la cargaison transportée et une réduction de 90% d'heures-hommes par rapport au plan manuel. Ces résultats sont considérés comme incomplets dans certains cas car le système ne tient pas compte, explicitement, du poids de l'essence transporté pendant chaque mission. Les auteurs reconnaissent que le *DMES* pourrait ne pas bien fonctionner lors de déploiements d'envergure, le temps de résolution devenant trop élevé.

En parallèle, Anderson et Ortiz (1987) proposent une procédure interactive appelée *AALPS* ou *Automated Aircraft Loading Planning System* dans laquelle ils exploitent les possibilités offertes par la combinaison des techniques de système expert et d'une interface graphique. Le système a pour tâches de générer automatiquement des chargements valides, de valider les plans définis par l'utilisateur et de modifier les plans existants. La procédure suit la série d'étapes suivante : trier la cargaison, remplir l'avion, équilibrer l'avion, valider les contraintes et générer des chargements types. Ce système a connu un succès auprès de la *XVIII Airborne Corps* de l'armée américaine mais son désavantage majeur est lié à son coût de développement, estimé à 10 millions de dollars US et à sa portabilité limitée. Heidelberg *et al.* (1998) proposent

une nouvelle version de *AALPS* avec plus de succès. Un algorithme heuristique se basant sur le problème de placement ou *bin packing problem* à 2-dimensions est développé. Les items sont supposés rectangulaires et la satisfaction des contraintes sur les emplacements des items détermine la faisabilité du chargement. L'algorithme commence par trier les items, détermine la section de l'avion à charger, et décide de la possibilité de placer le prochain candidat de la liste ou son successeur à l'endroit ou de laisser la place vacante. Une fois qu'un item est placé, une barrière formée de suite de segments permet de délimiter la prochaine portion de l'avion à charger. L'algorithme peut ajuster la barrière après qu'un item est chargé ou si aucun candidat de la liste n'est retenu. L'algorithme permet de "glisser" les items de l'avant vers l'arrière sans perturber le processus de chargement, mais les déplacements latéraux des items sont difficilement gérés. L'approche est revue par les professionnels du chargement et a été reconnue satisfaisante pour des gros plans de chargement par l'*U.S. Air Force* et l'armée américaine.

Une limite des méthodes proposées réside dans le fait qu'elles ne fournissent pas une garantie de la qualité de la solution. Notons aussi que les difficultés majeures associées à ces outils sont liés à leur version, laquelle consiste à utiliser un modèle théorique simplifié du problème de chargement (Martin-Vega, 1985).

Dans l'armée canadienne, Ng (1992) a proposé un modèle qui sélectionne, parmi un ensemble de chargements admissibles, définis au préalable, un sous-ensemble qui permettra d'atteindre les trois objectifs suivants : assurer le transport de tous les items en conformité avec les priorités de livraison données, minimiser le nombre d'avions-cargos *Hercules CC130* requis pour transporter les items et maximiser la capacité excédentaire en termes de surface ou de poids. Il a décrit un exemple en utilisant des données réelles (20 catégories d'items et 38 plans de chargement définis au préalable et testés). D'après la solution obtenue par le modèle, 110 chargements d'avions seraient nécessaires pour transporter tous les items ; si l'on compare aux 121

avions réellement utilisés lors de cette mission, la différence n'est pas négligeable. En effet, bien que le modèle soit une heuristique, il a néanmoins réduit le chargement des avions-cargos de 9%, comparé à la méthode manuelle et a épargné le coût de 240 heures de vol. Mais il y a de fortes limites dans l'utilisation des chargements standards car plusieurs items n'ont pas de dimensions standards. Sous sa forme actuelle, le modèle n'offre pas la possibilité de modifier les plans de chargement, mais peut être néanmoins utile pour obtenir une estimation du nombre d'avions nécessaires pour la planification.

Des études de cas montrent aussi que certains problèmes de chargement d'avions-cargos peuvent être résolus à l'optimalité ou du moins de façon satisfaisante. Ainsi, pour le compte d'une compagnie aérienne dont le compartiment des avions utilisés est divisé en cinq quais, Brosh (1981) formule le problème de maximiser la charge utile d'un avion comme un programme fractionnaire. Les contraintes non linéaires décrivant la relation entre la zone prédéfinie de l'enveloppe non convexe du CG et le poids du chargement ont été convexifiées par approximations linéaires. A partir d'une valeur initiale, l'auteur résout le programme linéaire correspondant. Si la solution se trouve à l'intérieur de l'enveloppe prédéfinie, la solution est acceptée. Sinon, on raffine l'approximation et un nouveau programme linéaire est résolu. L'auteur montre que la procédure converge vers une solution ϵ-optimal. L'auteur admet aussi que son algorithme exploite la structure particulière du cas étudié du point de vue théorique et informatique.

Des travaux similaires effectués par Thomas *et al.* (1998) pour le compte de *Federal Express*, une vaste compagnie de transport, méritent aussi d'être cités. Le problème consiste à optimiser le temps de chargement et de déchargement des conteneurs dans un *Airbus A300*, un avion léger conçu pour le transport de passagers, car le chargement de l'avion commence avant l'arrivée de tous les conteneurs. Les conteneurs sont restreints à des positions précises selon leur dimension, et des limites de poids

restreignent aussi ces conteneurs, une fois chargés, dans une zone déterminée. Par ailleurs, la charge utile globale de l'avion ne doit pas dépasser sa charge maximale permise et une mesure du CG de l'ensemble avion-conteneurs permet de valider la méthode de chargement. La formulation du problème est non linéaire et une approche de *B&B* est utilisée. La procédure se fait en deux phases. Dans un premier temps, on détermine un chargement réalisable. Si aucune affectation n'est possible, il faut enlever un ou plusieurs conteneurs. On répète cette procédure tant qu'on ne trouve pas une solution. Dans la seconde phase, on ajoute des contraintes de déchargement et des préférences aux items chargés pour mettre en valeur le chargement actuel. Les contraintes de déchargement reflètent le coût encouru en temps si l'item est déchargé à cause d'une infaisabilité éventuelle du chargement courant. Les contraintes de préférence se posent lorsqu'il y a plusieurs arrêts, de sorte que les items qui sont déchargés dans les premières destinations doivent être placés près de la porte. Si le plan est réalisable, l'algorithme s'arrête. Sinon, une méthode par essais et erreurs permet d'enlever un conteneur contraignant en commençant par le conteneur situé près de la porte et en continuant vers l'arrière de l'avion jusqu'à l'obtention d'une solution réalisable. L'algorithme effectue une recherche en profondeur pour trouver rapidement une solution locale et le coût de la solution se traduit par le nombre de retours en arrière ou *backtracking* effectués dans l'arbre de recherche. Les problèmes résolus comportent 800 colonnes, 300 rangées, et un total de 45 000 éléments non nuls. Les auteurs notent que le volume transporté est plus restrictif que son poids, de sorte que le concept standard qui consiste à placer les items les plus lourds près des ailes de l'avion fonctionne toujours assez bien. L'approche utilisée est assez générale et peut s'appliquer aux petits ou gros avions de transport, auxquels sont liés les contraintes sur les charges autorisées et le CG de l'ensemble. L'inconvénient de l'approche est qu'elle ne considère qu'un nombre limité de chargements, l'approche de génération de colonnes serait la plus adaptée.

Mongeau et Bès (2000) résolvent le problème de chargement de conteneurs dans un *Airbus A340-300*, un gros transporteur commercial, mais avec une fonction objectif

différente qui consiste à charger dans l'avion le plus de conteneurs possible tout en le balançant pour minimiser la consommation d'essence. Les contraintes permettent d'affecter un sous-ensemble de ces conteneurs à des compartiments fixes, lesquels sont soumis à des contraintes structurelles incluant la capacité par volume ou par poids que le compartiment peut supporter ou le poids maximal de l'avion, une fois chargé. Le problème est formulé comme un programme linéaire avec des variables binaires indiquant quel conteneur placer à quelle position. Le programme est résolu à l'optimalité par un algorithme de *B&B*. Des expériences numériques sur des problèmes réels et des problèmes construits à la main rapportent la viabilité de la méthode en pratique. Une extension intéressante proposée par les auteurs serait d'étudier l'impact des différentes variations de la fenêtre de faisabilité du CG. Ainsi, un compromis entre les deux objectifs contradictoires suivants, à savoir charger au maximum l'avion et optimiser la consommation d'essence, peut être possible. De plus, les auteurs considèrent aussi la possibilité d'implanter une procédure qui choisisse automatiquement une solution parmi celles qui permettent d'épargner des coûts, connaissant les revenus par tonnage de charge de fret, l'impact en termes de gain par centimètres de variation du CG et le coût courant de l'essence.

Dans ces études de cas, nous notons que les avions étudiés sont compartimentés et que les items ou conteneurs sont placés à des positions fixes. Les problèmes consistent donc à charger chaque position au mieux, suivant certaines contraintes. Le quai des avions *Hercules CC130* n'a pas ces contraintes. Les modèles linéaires en nombre entiers de chargement d'un avion sont intéressants car le problème à un avion est un sous-problème pouvant être appelé plusieurs fois et il a de fortes chances de trouver une solution optimale plus rapidement qu'un autre non linéaire.

A propos du problème de balancement d'un avion, Amiouny *et al.* (1992) présentent une heuristique destinée à charger des blocs, homogènes ou non, à l'intérieur d'un avion ou d'un camion, de manière à ce que le centre de gravité de ce dernier se trouve

le plus près d'un point cible p. Ils ont prouvé que le problème de balancement à une dimension est NP-complet. Ils se sont limités au problème de balancement de l'avion dans le sens de sa longueur au cours d'un seul chargement de tous les blocs. Leur heuristique est basée sur le principe du moment induit par un objet par rapport à un point de référence, dont le calcul correspond au produit du poids de l'objet par la distance entre son CG et le point de référence. La première étape de leur algorithme, appelé *Balance*, consiste à trier les items (blocs) par ordre croissant de densité. Ainsi, les blocs seront chargés par ordre croissant de densité en partant des extrémités de l'avion. Et pour ce faire, ils ont procédé par étapes : d'abord charger le bloc suivant, et ensuite transformer le problème en un problème similaire ayant un nouveau point cible p et un bloc de moins à charger. Le nouveau point cible est établi de telle sorte qu'un parfait balancement des blocs restants contrebalance le moment induit par le bloc chargé. Ils ont montré que leur méthode produit un chargement pour lequel le CG de l'ensemble des blocs est, soit à l'intérieur de $\frac{1}{2}l_{max}$ de p soit le plus près possible de p (l_{max} désigne la longueur du bloc le plus long). Par ailleurs, ils ont effectué des comparaisons entre l'algorithme *Balance* et d'autres algorithmes tels *Permutation* et *2-interchange*. Bien que *Permutation* exige environ le même temps de calcul, les résultats démontrent qu'il est moins efficace que *Balance*. L'algorithme *2-interchange* fournit de bons résultats mais exige un temps de calcul plus grand.

Mathur (1998) propose une alternative à l'heuristique d'Amiouny *et al.*. Son algorithme se base sur l'approximation du problème comme un problème de sac à dos. Comparé à l'algorithme *Balance* d'Amiouny *et al.*, il a la même complexité de résolution, mais avec une meilleure performance dans le pire cas. En général, les résultats numériques montrent que son approche réagit mieux sur les problèmes générés aléatoirement.

Notre étude est plus particulière que les problèmes ou les études de cas ou encore les approches heuristiques pour le balancement d'avion présentés dans cette revue de la

littérature. En effet, nous désirons optimiser le nombre d'avions et les distances parcourues. Dès lors, nous nous heurtons à une grande difficulté dans la modélisation de ALP : outre le lieu de collecte des items, nous devons tenir compte aussi de leur longueur, de leur poids et de leur ordre de priorité. Ces items ne sont pas décomposables, et dans un chargement réalisable, ils ne sont pas forcément tassés ou mis bout à bout pour satisfaire aux contraintes opérationnelles de chargement. De plus, l'emplacement de ces items dans l'avion est également soumis à une autre contrainte : on doit les mettre soit sur le quai, soit sur la rampe, chevaucher les deux n'est pas possible. Cette contrainte d'emplacement est valable pour chaque item, et pour compliquer encore le problème, les contraintes collectives doivent être satisfaites. Par exemple, remarquons que par rapport à sa borne inférieure un chargement réalisable courant peut devenir non réalisable à cause des items chargés ultérieurement, et l'inverse est aussi vrai, c'est-à-dire qu'un chargement non réalisable peut devenir réalisable dépendamment des items à venir. En somme, c'est la contrainte de borne inférieure du CG qui est la plus difficile à gérer. Par ailleurs, pour un même type d'item, telles les jeeps par exemple, la demande peut être plus d'une unité, ainsi, le fait de permuter deux jeeps quelconques, dans deux chargements distincts, contenant respectivement une jeep, mène à une autre solution optimale. Cette difficulté est connue comme celle de la symétrie du problème.

Récemment, dans un article traitant des problèmes déterministes de tournées de véhicules avec fenêtres de temps et d'horaires d'équipage, Desaulniers *et al.* (1998) proposent un modèle unifié pour résoudre la plupart de ces problèmes. Un problème générique apparaissant dans ces applications a été identifié : couvrir à moindre coût un ensemble de tâches en choisissant, dans un graphe, des chemins respectant des contraintes d'opération. De façon plus simple, et dans le cas des problèmes de tournées de véhicules, les tâches peuvent représenter des clients à visiter, des trajets d'autobus, des vols d'avion, etc. et dans le cas des problèmes d'horaires d'équipage, elles

sont des trajets d'autobus ou des vols d'avion entre les points où l'on peut remplacer le personnel. Les chemins représentent soit des itinéraires qu'il faut determiner pour effectuer toutes les tâches, soit des suites de tâches effectuées par un équipage (journées de travail ou rotations de travail). Les contraintes d'opération ou les restrictions sur ces chemins portent entre autres, sur la capacité des véhicules ou sur la distance parcourue pour les cas de tournées de véhicules, sur les règles provenant de la convention de travail des employés pour le cas d'horaires. Si les chemins sont bien construits dans les applications étudiées, la résolution du problème générique dépend du choix des sous-ensembles qui couvrent toutes les tâches programmées. Les choix possibles font alors que ces problèmes sont combinatoires. Mathématiquement, le modèle proposé est une formulation non linéaire d'un problème de flot multi-commodités avec des variables entières et des variables de ressource. C'est une extension d'une formulation du problème du voyageur de commerce multiple avec variables de ressource ou *m-TSPR (multiple-Traveling Salesmen Problem with Resource variables)*. Ce dernier consiste à visiter un ensemble de clients exactement une seule fois, au coût minimal au moyen d'une flotte hétérogène de véhicules. Au lieu de parler des contraintes du problème sur les chemins, les auteurs font appel à des ressources qui sont des commodités consommées dans le réseau. Les consommations sont transférées sur les arcs du réseau et sont cumulées sur les arcs d'un chemin, tandis que les restrictions sur les ressources sont imposées sur les nœuds et sont vérifiées et mises à jour à chaque nœud au moyen d'une fonction appelée fonction de prolongation.

La version d'*ALP* qui nous intéresse consiste à transporter, à moindre coût, un ensemble d'items au moyen d'une flotte hétérogène d'avions dont chaque chargement et/ou dont les routes sont toutefois soumis à des restrictions ou des contraintes opérationnelles de chargement. De ce fait, le problème générique décrit par Desaulniers *et al.* (1998) s'apparente à *ALP*. En effet, les tâches correspondent aux items à transporter, les véhicules sont les avions et un chemin correspond à une suite d'items

chargés dans un avion et est soumis à des contraintes d'opération que nous avons décrites comme des restrictions opérationnelles de chargement. Des ressources sont conçues pour "capturer" ces restrictions, entre autres, la longueur et le poids des items chargés, le centre de gravité de l'ensemble des items chargés ou les villes déjà visitées. Nous proposons alors de formuler *ALP* comme un modèle de tournées de véhicules, d'autant plus que les problèmes de tournées de véhicules de grande taille sont en général résolus d'une façon assez efficace avec une méthode de séparation et d'évaluation progressive faisant appel à une technique de décomposition de Dantzig-Wolfe ou méthode de *branch-and-price.*

Toutefois Barnhart *et al.* (1998) font état de difficultés énormes à surmonter dans l'application de cette méthode. D'abord, le branchement standard sur les variables peut ne pas fonctionner car fixer certaines variables peut détruire la structure du sous-problème. Ensuite, la résolution du *LP* et du sous-problème jusqu'à l'optimalité ne sont pas recommandées dans certains problèmes pour lesquels des règles sont appliquées pour gérer l'arbre de recherche. Le nombre d'itérations requises pour montrer l'optimalité du *LP* ou *tailing-off effect* est une difficulté type de cette approche (Vanderbeck, 1994).

Nous allons traiter toutes les difficultés citées ci-dessus et résoudre une part importante du problème. Le succès de la résolution de *ALP* dépend alors de la conception d'un réseau approprié et de la transformation des contraintes opérationnelles du problème par des contraintes mathématiques, pour atteindre un niveau de réalisme de modélisation, de l'obtention d'une bonne approximation de la fermeture convexe des solutions réalisables et d'une méthode de branchement efficace qui défait la symétrie du problème et/ou élimine un grand nombre de variables.

Dans les sections suivantes, la méthode de séparation et évaluation est décrite et la méthode de génération de colonnes sera détaillée par la suite. Enfin, une vue assez

rapide des autres alternatives de résolution du modèle, rencontrées dans la littérature, termine ce chapitre.

1.2.1 La méthode de séparation et évaluation progressive ou *branch-and-bound*

La méthode de séparation et évaluation ou *branch-and-bound*, notée $B\&B$, est utilisée pour résoudre de façon exacte des problèmes d'optimisation combinatoire appartenant à la classe des problèmes $\mathcal{NP}$-difficiles. C'est une technique d'énumération implicite qui consiste à décrire l'ensemble des solutions sous la forme d'une arborescence et à la parcourir intelligemment. A une étape de la méthode, un sous-ensemble correspondant à un sommet pendant de l'arborescence est choisi et ce sous-ensemble sera divisé ou partitionné en des sous-ensembles de plus en plus petits de façon à isoler une solution optimale dans l'un de ces sous-ensembles. Une évaluation de ces sous-ensembles permet d'éliminer des branches complètes de l'arborescence et de localiser une solution optimale.

Dans le cas des programmes linéaires en nombres entiers ou *PLNE*, la méthode du $B\&B$ commence par résoudre une relaxation du programme linéaire, par exemple en enlevant les conditions d'intégralité sur les variables. Cette évaluation est toujours inférieure ou égale à la valeur de la solution optimale du *PLNE*, dans le cas d'un problème de minimisation, puisque toute solution du *PLNE* est une solution de la relaxation. Par conséquent, si la solution optimale de la relaxation linéaire est entière, alors c'est la solution optimale du *PLNE*. Sinon, on effectue une opération de séparation qui permettra de construire l'arborescence et une règle doit être définie pour se déplacer à l'intérieur. On évalue ensuite les nouveaux nœuds ou sous-ensembles et, éventuellement, on élague ceux qui sont inutiles. Un nœud peut être élagué si :
- le programme linéaire n'est pas réalisable ;

- la valeur de la solution est supérieure à la valeur de la meilleure solution réalisable trouvée, dans ce cas, il est inutile de continuer la recherche dans la sous-arborescence enracinée en ce nœud, puisque les bornes inférieures obtenues sont strictement croissantes suivant la profondeur de l'arbre du $B\&B$;
- la solution est entière, donc réalisable.

L'algorithme s'arrête quand on n'a plus de nœud à évaluer.

Il y a trois types de nœuds dans l'arbre du $B\&B$ pendant le déroulement de l'algorithme, le nœud courant qui est en train d'être évalué, des nœuds actifs qui sont dans la liste des nœuds qui doivent être traités, et des nœuds inactifs qui ont été élagués au cours du calcul. Quand la liste des nœuds actifs est vide, la meilleure solution réalisable obtenue est la solution optimale du problème. On cherchera donc, pour réduire la combinatoire, à définir la plus petite arborescence possible. Le choix de la stratégie de sélection du nœud actif à traiter et le choix de la variable de branchement influent lourdement sur la taille de l'arborescence. Ces derniers points seront plus détaillés lors de l'application de la méthode.

1.2.2 La méthode de génération de colonnes

Plusieurs problèmes d'optimisation, entre autres *ALP*, peuvent être formulés comme un problème de partitionnement d'ensemble ou *Set Partitioning Problem (SPP)* : étant donné un ensemble d'éléments et une collection de sous-ensembles auxquels on associe un coût, le problème consiste à sélectionner des sous-ensembles de telle sorte que chaque élément soit utilisé un nombre précis de fois et que le coût total des sous-ensembles sélectionnés soit minimisé. Ce modèle s'écrit sous la forme d'un programme linéaire en nombres entiers où chaque sous-ensemble correspond à une colonne et chaque élément est associé à une rangée. Dans le contexte de *ALP*, les éléments sont les items, un sous-ensemble d'éléments correspond à un chargement

réalisable ou encore *chalk* dans le jargon militaire, et la contrainte liée à chaque item est qu'il soit chargé un nombre précis de fois. Pour chaque colonne, soit θ_e la variable définie comme le nombre de fois que le chargement e est utilisé dans la solution et soit c_e le coût associé à ce chargement. Le programme résultant s'écrit :

$$min \sum_{e \in \Omega} c_e \theta_e \tag{1.1}$$

$$\sum_{e \in \Omega} a_{we} \theta_e = q_w, \qquad \forall w \in N \tag{1.2}$$

$$\theta_e \in \mathbb{Z}^+ \tag{1.3}$$

où Ω est l'ensemble de *tous* les chargements possibles, N est l'ensemble de tous les items à transporter, a_{we} est défini comme étant le nombre de copies de l'item w se trouvant dans le chargement e et q_w est égal au nombre de copies désirées.

Notons que toutes les contraintes opérationnelles sont supposées respectées car Ω contient tous les chargements réalisables et uniquement des chargements réalisables. De plus, une solution provenant de cette formulation sous forme de partitionnement ne décrit pas complètement une solution à notre problème car la séquence de chargement dans un avion n'est pas donnée. Si, au lieu de considérer le problème classique de partitionnement, nous remplaçons le "=" de la contrainte de demande par un "$\geq$", nous avons la formulation sous forme de problème de recouvrement ou *Set Covering Problem* ou *SCP*. Dans ce cas, la contrainte pour chaque item serait alors de charger l'item *au moins* autant de fois que le nombre fixé. En général, lorsque nous avons le choix entre la formulation *SCP* et *SPP*, la première est préférée car, selon Barnhart *et al.* (1998), la résolution de sa relaxation linéaire ou *LP* est plus stable et donc plus facile à résoudre, et de plus, il est aisé de construire une solution entière réalisable à partir d'une solution du *LP*. Mais pour *ALP*, la première formulation n'est pas adaptée. En effet, décider de ne plus charger un item contenu dans un chargement réalisable peut rendre ce chargement non-réalisable à cause de la contrainte du CG,

de ce fait, une solution optimale du *SCP* n'est pas toujours une solution optimale du *SPP*.

Pour résoudre le *SPP*, nous avons deux choix : soit utiliser une formulation dont les sous-ensembles réalisables vont être définis implicitement par un ensemble de contraintes, soit utiliser une formulation qui comprendra explicitement tous les sous-ensembles réalisables. Dans de nombreuses applications réelles, la deuxième alternative a plus de chance d'aboutir à une solution optimale. Cependant, lorsque la taille du problème à traiter est assez grande, elle peut devenir impraticable. Par exemple, si 300 items doivent être transportés par des avions identiques pouvant charger jusqu'à 5 items par vol, le nombre de colonnes pour décrire le problème serait de l'ordre de 2×10^{10}. En utilisant, par contre, une représentation partielle de cette formulation, c'est-à-dire une formulation contenant un nombre limité de colonnes, nous pouvons contourner la difficulté. L'idée serait alors de travailler sur un ensemble de colonnes de taille réduite et d'être capable de prouver ou non que la solution optimale du problème *SPP* est décrite par une combinaison de ces colonnes. Pour ce faire, nous devons disposer d'un générateur de colonnes qui, vu la solution courante et les colonnes associées correspondantes, propose une ou plusieurs colonnes qui amélioreront cette solution ou affirme l'optimalité de la solution. La technique que nous avons décrite est appelée la méthode de génération de colonnes, le générateur de colonnes sera appelé le sous-problème ou *SP* et le modèle linéaire comportant l'ensemble restreint de colonnes sera appelé le problème maître restreint ou *PMR*.

Pour trouver la solution optimale du problème original, on applique la méthode de "*branch-and-price*". C'est une généralisation de la méthode de *B&B* dans laquelle l'algorithme de génération de colonnes est appelé pour générer des nouvelles colonnes, au besoin, et calculer une borne inférieure du *LP* à chaque nœud de l'arbre de recherche. Le branchement peut être effectué quand le générateur de colonnes ne trouve plus de colonnes améliorantes pour le *PMR* et que la solution du problème relaxé ne

satisfait pas aux conditions d'intégralité. Au départ de l'algorithme de génération de colonnes, le *PMR* est constitué d'un ensemble de colonnes qui permet de construire une solution réalisable du problème relaxé. Le *PMR* est résolu par l'algorithme primal du simplexe et son rôle est de trouver la meilleure solution à partir des colonnes courantes ainsi que les variables duales associées à chaque contrainte. Ces variables duales sont utilisées par le *SP* pour identifier les colonnes de coût réduit négatif. Si des colonnes intéressantes sont trouvées, elles sont ajoutées dans le *PMR* et ce dernier est réoptimisé. Sinon, nous considérons que l'optimalité a été atteinte, c'est-à-dire que la solution du *LP* est obtenue. Notons que c'est le rôle du *SP* de veiller à ce que les colonnes générées soient toutes réalisables, et donc le *SP* supporte une part importante de la difficulté du problème.

Rappelons que les colonnes sont les variables du *PMR* et selon Barnhart *et al.* (1998), il y a plusieurs raisons de considérer une formulation avec un nombre très grand de variables. D'abord, la borne obtenue par la relaxation linéaire d'une formulation compacte peut être une mauvaise approximation de la valeur optimale entière du problème. De ce fait, la résolution du programme par l'algorithme de *B&B* standard ne permet pas de résoudre une instance de grande taille du problème. Une reformulation qui implique un assez grand nombre de variables peut améliorer cette relaxation. En outre, la formulation compacte présente beaucoup de symétrie et cela risque de faire défaut à la méthode de *B&B* standard car c'est à peine si un changement s'opère dans le problème après un branchement. Une reformulation peut aussi éliminer cette symétrie. Un autre avantage est que la génération de colonnes offre une décomposition naturelle, à savoir le problème maître ou niveau agrégé et le sous-problème ou niveau désagrégé, elle permet aussi d'incorporer des contraintes additionnelles. Enfin, nous retenons notre attention sur les récents succès de cette technique capable de résoudre des problèmes réels (voir Desrosiers *et al.* , 1995).

1.2.3 Autres méthodes pour trouver une bonne borne inférieure

La méthode de la relaxation lagrangienne est une alternative pour obtenir une bonne borne inférieure de *ALP*. C'est une approche par décomposition qui tire profit de la formulation composée d'une fonction objectif et des contraintes séparables par commodité, formant une structure diagonale en plusieurs blocs avec des contraintes liantes. La méthode isole certaines contraintes dans des sous-problèmes résolus au moyen d'algorithmes spécialisés, entre autres, l'algorithme de sous-gradients. Cette méthode est simple et facile à implanter pour des applications spécialisées. Cependant, pour de larges instances, l'algorithme peut être lent et peut causer quelques problèmes de convergence, voir Barnhart *et al.* (1998). Par ailleurs, il est bien connu que le problème dual lagrangien obtenu après la relaxation des contraintes liantes fournit une relaxation équivalente, c'est-à-dire donnant la même borne inférieure que le problème maître de la décomposition de Dantzig-Wolfe. Geoffrion (1974) a montré, dans ce cas, que les deux problèmes sont duaux. Cependant, lorsque la taille du problème est très grande, l'obtention des solutions primales constitue un avantage important de la méthode de décomposition de Dantzig-Wolfe par rapport à la relaxation lagrangienne. Pour plus de détails sur la comparaison entre les deux méthodes, le lecteur peut consulter Barnhart *et al.* (1998).

D'autres alternatives, telles les méthodes de faisceaux, (voir Lemaréchal, 1989), ont été utilisées récemment pour les problèmes d'optimisation non différentiables. Elles apparaissent très prometteuses mais n'ont pas été testées sur des problèmes de grande taille.

La méthode de plans sécants ou *branch-and-cut* est aussi une approche qui a retenu l'attention des chercheurs ces dernières années pour la résolution des problèmes mixtes ou entiers (*MIP*) de grande taille. C'est une généralisation de l'algorithme de

B&B et l'idée générale de la méthode est d'améliorer la description polyédrale de l'espace des solutions en "coupant" les parties non entières de l'espace des solutions du problème relaxé avant, éventuellement, que le branchement s'effectue. Nous donnons un aperçu de cette méthode, le lecteur peut consulter Nemhauser et Wolsey (1988) pour une présentation plus détaillée.

Il est bien connu que la résolution du *LP* donne une solution entière si on arrive à obtenir des descriptions de l'enveloppe convexe de toutes les solutions entières du problème linéaire par des contraintes. Une fois que la borne inférieure est calculée lors de la relaxation du *LP*, si la solution n'est pas entière, la meilleure façon d'améliorer cette borne est d'introduire des contraintes dans le problème. Les contraintes appelées coupes sont les plus recherchées. Une coupe est une inégalité valide c'est-à-dire une inégalité qui est satisfaite par toutes les solutions réalisables. Ces inégalités ne sont pas présentes dans la formulation courante et elles ne sont pas satisfaites par toutes les solutions du *LP*, le cas échéant, on les appelle coupes violées. Ces dernières peuvent être ajoutées à la relaxation du *LP*, et de ce fait, améliorent la borne de celle-ci. En d'autres termes, la formulation courante a été modifiée car la région réalisable du *LP* courant est rendue plus petite, par contre la région contenant l'ensemble des solutions réalisables du *MIP* reste la même. Ainsi, si une ou plusieurs coupes violées sont identifiées, elles sont ajoutées au *LP* pour couper la solution non réalisable, et le *LP* est réoptimisé. Les branchements interviennent quand aucune coupe n'est trouvée. Les deux procédures, à savoir la recherche de coupes violées et le branchement, sont appliquées à chaque nœud de l'arbre de recherche d'un algorithme de *B&B*.

Pour clore ce chapitre, mentionnons que l'algorithme de plans sécants donne aussi la même borne inférieure que l'algorithme de génération de colonnes ou la méthode de relaxation lagrangienne (Vanderbeck, 1994). Même si la méthode garantit une convergence après un nombre fini d'itérations, cette procédure peut être très longue. Cela est dû à la génération de coupes faibles ne coupant qu'une petite partie du domaine

fractionnaire de la solution. Ces coupes alourdissent le programme et ralentissent sa résolution. Les coupes fortes sont celles qui coupent profondément le domaine des solutions fractionnaires et elles sont particulières à la structure du problème considéré. Elles sont difficiles à trouver, aussi est-on obligé de chercher un compromis entre la profondeur des coupes et le temps nécessaire pour les trouver. Plusieurs coupes sont proposées dans la littérature pour différents problèmes d'optimisation combinatoire, en particulier les coupes compatibles avec la méthode de génération de colonnes, sur lesquelles nous reviendrons un peu plus loin.

CHAPITRE 2 : LE MODÈLE DE BASE

2.1 Introduction

Dans ce chapitre, nous présentons le modèle de base du problème de chargement des avions-cargos. Nous nous intéressons au problème de chargement des avions-cargos où tous les items se retrouvent en un seul point de collecte et pour une seule destination de livraison.

Nous proposons le développement d'un modèle qui construit des plans de chargements réalisables qui minimisent le coût total. Dans ce cas particulier, la distance entre le point de collecte et le point de livraison est fixe. On peut ne pas en tenir compte dans le calcul des coûts. Quant aux coûts liés à l'essence, dû au fait qu'on n'a qu'un seul type d'avion et que la distance est fixe, ils ne sont pas considérés. Ne seront alors considérés que les coûts de sortie d'un avion. La fonction objectif consiste à minimiser le nombre d'avions requis pour transporter tous les items. Nous supposons que chaque exemplaire d'item à transporter représente un item spécifique et que la demande de chargement est de une unité pour chaque exemplaire. Ainsi, chaque avion part d'une base où il est chargé d'items et y rentre après avoir effectué sa livraison. Chaque item doit être chargé exactement une fois dans un avion en respectant certaines restrictions, entre autres les priorités d'envoi. Enfin, les contraintes opérationnelles de chargement de chaque avion doivent être satisfaites.

Nous présentons la formulation mathématique du problème de chargement. Une formulation a été proposée par Desaulniers *et al.* (1998) pour traiter des problèmes déterministes de tournées de véhicules avec fenêtres de temps et d'horaires d'équipage. Le problème générique apparaissant dans ces applications est défini sur un graphe.

Les nœuds de ce graphe représentent les tâches à exécuter, par exemple des clients à servir, et les arcs représentent la possibilité de faire suivre deux tâches dans la même tournée. La construction des chemins sous contraintes d'opération est déterminante pour exploiter au mieux l'approche.

La première section de ce chapitre étudie un exemple tiré du problème de Ng (1992). La résolution de l'exemple illustre les différentes étapes de la construction d'un réseau de chargement sur lequel les équations du modèle mathématique se basent. Ces étapes constituent un prétraitement pour l'étude d'un problème réaliste car elles mettent en valeur une procédure de réduction de la taille en termes d'arcs ou de nœuds du réseau ou encore l'élimination de certaines symétries du problème. Dans la deuxième section, les équations du modèle mathématique sont posées et les variables de ressources sont formulées en détails. Nous montrons, à la section trois, comment traiter quelques cas spéciaux du problème, entre autres les priorités d'envoi des items, le transport des passagers ou des items particuliers comme les hélicoptères ou les remorques. Dans la quatrième section, nous abordons l'algorithme utilisé pour résoudre ce modèle. Enfin, dans la dernière section, l'efficacité de l'approche est montrée sur un ensemble de tests numériques.

2.2 Étude d'un exemple

Supposons que nous ayons à transporter *entre deux points* les huit items du tableau 2.1. Dans ce tableau, on donne la quantité q_w, la longueur l_w (en pouces), le poids m_w (en livres) et les priorités d'envoi $Prio_w$ selon le type w de l'item.

La longueur du quai est de 492 unités et celle de la rampe 132. Le quai peut supporter jusqu'à 195 900 lbs et la rampe seulement 5 000 lbs. Le problème que nous essayons

Tableau 2.1 – Données pour l'étude de certaines symétries

w	q_w	l_w	m_w	$Prio_w$	Description de l'item
1	1	267	17 000	2	Medium Logistics Vehicle Wheeled 2 $\frac{1}{2}$ ton
2	3	232	24 800	3	Armoured Personel Carrier M113.
3	2	109	1 100	2	$\frac{1}{4}$ Ton Trailer.
4	2	147	2 900	1	$\frac{3}{4}$ Ton Trailer

de résoudre revient à chercher des sous-ensembles de l'ensemble des chargements admissibles. Un chargement est admissible s'il respecte les contraintes opérationnelles de chargement même si un ou plusieurs items apparaissent plusieurs fois dans le chargement. Nous supposons que pour un chargement donné, l'emplacement exact d'un item dans l'avion n'est pas encore complètement déterminé, mais en revanche nous nous assurons de la faisabilité du chargement. Nous supposons aussi qu'un chargement composé de l'item 1 puis de l'item 2 est différent d'un chargement composé de l'item 2 puis de l'item 1 à cause du CG.

D'après ces données, on peut composer huit chargements comportant un seul item, 64 chargements comportant deux items, 512 comportant trois items et 1 024 comportant quatre items car un seul exemplaire de l'item de type 3 peut être mis sur la rampe. Nous avons un total de 1 608 chargements possibles, et en général lorsque la taille du problème est assez grande, l'espace de recherche peut contenir, a priori, des milliards de chargements admissibles. Par ailleurs, on peut réduire ce nombre en éliminant certaines symétries et certaines incohérences sur ces chargements qui sont indésirables et qui détériorent la performance de l'algorithme utilisé. Il s'agit alors de construire une procédure ou des prétraitements capables de générer les bons chargements.

2.2.1 Esquisse d'un réseau de chargement

Pour procéder à une première réduction du nombre de solutions réalisables, nous pourrions imposer un rang entre les exemplaires d'items de même type et, de ce fait, à toute solution quelconque, nous pouvons associer une solution respectant le rang établi pour les différents exemplaires d'items en réordonnant, au besoin, la solution courante. Il est possible d'effectuer une autre réduction en gérant de manière efficace les liens entre les exemplaires de chaque item quelconque. Cette procédure sera explicitée plus en détail à partir du réseau test.

Remarquons que, pour n'importe quel chargement réalisable, on peut toujours établir un ordre de chargement pour les items qui le composent. Un chargement est alors considéré comme une suite ordonnée d'items. Ainsi, le chargement ($item3\_1, item3\_2, item4\_2$) comporte trois items spécifiques. La première position est occupée par le premier exemplaire de l'item 3, la deuxième position par le second exemplaire du même item et la troisième position par le second exemplaire de l'item 4. Nous remarquons aussi qu'à partir des données sur les items, une borne supérieure sur le nombre d'items dans un chargement quelconque peut être calculée. Même si ce nombre est assez élevé, on ne peut encourir aucune perte de généralité. Dans l'exemple présent, comme la longueur est la contrainte la plus contraignante lorsqu'on charge sur le quai et en considérant les plus petits items, on pourra charger au plus deux items du type 3 et un item du type 4. Donc, au plus trois items sur le quai. Pour la rampe, on pourra y placer au plus un item du type 3. Ainsi, le nombre maximum d'items chargés est de 4 pour cet exemple.

Ici, nous avons trois priorités d'envoi. Afin de respecter ces dernières, nous devons acheminer à destination les items de priorité 1 avant les items de priorité 2 ou 3, et ainsi de suite. Afin de contourner cette difficulté, il suffit d'assurer que chaque

avion ne contienne que des items de priorités égales ou successives. En effet, une fois que tous les plans de chargement sont trouvés, il nous reste à exécuter les plans en commençant par ceux contenant uniquement des items de priorité 1, ensuite par ceux contenant les priorités 1 et 2, et après les plans contenant uniquement les priorités 2, et ainsi de suite jusqu'à ce que tous les plans soient utilisés. Dès lors, nous serons assurés que les items de priorité 3 (item 2) ne pourront pas arriver à destination avant ceux de priorité 1 (item 4) car ils ne seront jamais dans le même avion.

Notre but est de chercher des chargements réalisables. Nous le ferons au moyen d'un réseau. Considérons alors le réseau "naïf" suivant. Les nœuds correspondent aux items à transporter ainsi qu'à la position qu'ils occupent dans le chargement. Dans notre exemple, nous avons vu qu'au plus trois items peuvent être chargés sur le quai et au plus un sur la rampe. Nous avons alors trois séries de huit nœuds pour le quai et une série de deux nœuds pour la rampe, puisque seuls les deux items de type 3 peuvent y aller. Un nœud *source* correspondant au début du chargement et un nœud *puits* correspondant à la fin du chargement complètent l'ensemble des nœuds de ce réseau.

Un chargement correspond à un chemin dans ce réseau. Ce chemin débute au nœud *source* et termine au nœud *puits*. Les arcs de ce réseau indiquent quel est l'item suivant à charger. Il y a un arc entre le nœud *source* et chacun des nœuds de la première série d'items sur le quai. Un chemin utilisant un de ces arcs spécifie le premier item chargé sur le quai. Un autre arc ayant son origine au nœud *source* et se terminant au nœud *puits* correspond au chemin représentant le chargement vide. Pour chaque nœud de la première série, il y aura onze arcs partant de ce nœud. Huit de ces arcs aboutissent vers chacun des huit nœuds de la deuxième série, deux de ces arcs aboutissent aux nœuds associés aux items sur la rampe et le dernier termine au nœud *puits*. Un chemin passant par un des huit premiers arcs identifie l'item placé en deuxième position. Si le chemin emprunte un des deux suivants, cela indique que le deuxième item est placé sur la rampe. Et si ce chemin passe par l'arc terminant au

puits, alors il n'y aura qu'un seul item dans ce chargement. Une série d'arcs similaires est issue de la deuxième série de nœuds. Pour la troisième et dernière série de nœuds du quai, il n'y aura que trois arcs issus de chaque nœud, deux de ceux-ci arrivant aux nœuds de la rampe et l'autre au nœud *puits*. Finalement, un seul arc est issu de chacun des nœuds de la rampe et ces arcs aboutissent au nœud *puits*. Dans le cas où on considère des priorités, on élimine tous les arcs reliant des items de priorités incompatibles.

Un coût est défini pour chaque arc du réseau. Ce coût est chargé pour chaque unité de flot qui passe par l'arc. Tous les arcs issus du nœud *source* ont un coût de 1, à l'exception de l'arc $(source, puits)$ qui, comme tous les autres arcs du réseau, a un coût nul.

2.2.2 Élimination de certaines symétries du problème

La définition et la gestion des contraintes du réseau ne seront abordées que dans les prochaines sections. Pour l'instant, nous supposons que le réseau "naïf" décrit ci-dessus peut générer des chargements réalisables c'est-à-dire respectant les contraintes opérationnelles de chargement et les contraintes d'envoi des items. Notre but est d'améliorer ce réseau, en termes de taille et de colonnes générées, en étudiant en détail les connexions entre deux nœuds quelconques. Comme nous n'avons pas encore de moyens de contrôler les connexions pour les séries de trois nœuds et plus, de la même façon qu'un arc contrôle la connexion entre deux nœuds, nous devons nous attendre à ce que le réseau génère des chargements où le même item revient plus d'une fois. Dans ce cas, la méthode de *B&B* éliminera ce chargement. La contrainte de CG est relaxée pour l'instant et l'ordre des items dans un chargement n'intervient pas. Par contre, l'item chargé sur la rampe sera chargé après tous les autres.

Notons par y_i la variable représentant le nombre de fois que nous utilisons le plan i de chargement, i est un indice pris dans l'ensemble de tous les plans de chargements possibles. Lors de la relaxation linéaire du problème, huit variables sont retenues. Chacune de ces variables est décrite sous forme de colonne dont les coefficients ne sont pas tous nuls par rapport à la rangée de l'item correspondant dans le tableau 2.2. La notation *item2_3* fait référence au troisième exemplaire du deuxième type d'item. La contrainte de chargement de cet item est décrite par la rangée correspondante. Par exemple, la variable y_2 est composée de trois items dont l'item *item1_1*, l'item *item4_1* et l'item *item3_1*. Pour ce dernier, la ligne de l'item correspondant sera marquée par une astérisque dans le tableau pour signaler que l'item est chargé sur la rampe.

Tableau 2.2 – Première relaxation linéaire

min	y_1	$+ y_2$	$+ y_3$	$+ y_4$	$+ y_5$	$+ y_6$	$+ y_7$	$+ y_8$		
		$+ y_2$			$+ y_5$				$= 1$	(item1_1)
			$+2\ y_3$				$+ y_7$		$= 1$	(item2_1)
						$+ y_6$	$+ y_7$		$= 1$	(item2_2)
						$+ y_6$		$+2\ y_8$	$= 1$	(item2_3)
		$+ y_2^*$		$+ y_4^*$		$+ y_6^*$			$= 1$	(item3_1)
							$+ y_7^*$	$+ y_8^*$	$= 1$	(item3_2)
	$3\ y_1$	$+ y_2$							$= 1$	(item4_1)
				$+3\ y_4$	$+ y_5$				$= 1$	(item4_2)
	$y_1,$	$y_2,$	$y_3,$	$y_4,$	$y_5,$	$y_6,$	$y_7,$	y_8	≥ 0	

La relaxation linéaire du problème est donnée par le tableau 2.2. Une solution optimale est : $y_1 = \frac{1}{6}, y_2 = \frac{3}{6}, y_3 = \frac{1}{6}, y_4 = \frac{1}{6}, y_5 = \frac{3}{6}, y_6 = \frac{2}{6}, y_7 = \frac{4}{6}, y_8 = \frac{2}{6}$ et $y_i = 0$ pour les autres chargements possibles. La valeur de la fonction objectif est de $\frac{17}{6} = 2,83$. Bien que cette solution ne soit pas réalisable, car la notion de fraction de voyage n'a pas de sens dans la réalité, elle nous renseigne sur la borne inférieure d'une solution réalisable lorsqu'elle existe.

Nous allons améliorer le réseau décrit ci-dessus par une réduction de sa taille en termes de nœuds et d'arcs, par une diminution du nombre de chargements admissibles générés sans pour autant exclure des solutions potentielles et enfin par une

augmentation de la borne inférieure de la relaxation du problème. Dans cet exemple, le plan de chargement associé à y_4 charge trois fois l'*item4_2* sur le quai et l'*item3_1* une fois sur la rampe. De toute évidence, ce plan de chargement est infaisable mais il apparaît dans la solution car il a été généré par le réseau décrit ci-dessus, parmi tant d'autres, afin de rendre la fonction objectif minimale. En interdisant les plans de chargement de ce genre, nous n'excluons aucune solution réalisable, et nous augmentons la borne inférieure. Comme la contrainte : *aucun exemplaire d'item ne peut apparaître plus d'une fois dans l'avion* élimine des plans indésirables. Elle est par contre plus complexe à réaliser. Nous allons donc nous contenter d'une version moins restrictive. Ainsi, en ajoutant la contrainte suivante : *aucun exemplaire d'item ne peut apparaître deux fois de suite sur le quai*, les chargements associés aux variables y_1, y_3, y_4, y_8 ne sont plus acceptés. Ces contraintes sont prises en compte en éliminant les arcs horizontaux de la Figure 2.1 (a) du réseau. Dans cette figure, les connexions entre les exemplaires d'un *même type d'item* situés sur deux séries consécutives dans le réseau original sont dessinées. Remarquons aussi que les chargements *(item2_1, item2_2,item3_1)* et *(item2_2,item2_1,item3_1)* sont identiques à tout point de vue. Nous apportons donc les modifications suivantes au réseau de façon à ne retenir que le premier chargement. En fait, on voudra que lorsque deux items de même type se suivent dans le chargement alors le numéro du premier devra être inférieur au numéro du second. Les arcs obliques qui se dirigent du bas vers le haut sur la Figure 2.1 (a) sont retirés du réseau.

Au lieu de retirer certains arcs du réseau, nous allons procéder autrement. Nous regroupons les items de même type qui sont de même longueur, de même poids et de même priorité. En ajoutant deux nœuds intermédiaires représentant respectivement l'entrée et la sortie dans le groupe (Voir Figure 2.1 (b)), et en opérant quelques changements sur les connexions entre les nœuds, nous réduisons le nombre d'arcs du réseau précédent et nous satisfaisons les contraintes imposées. Chaque nœud d'entrée d'un

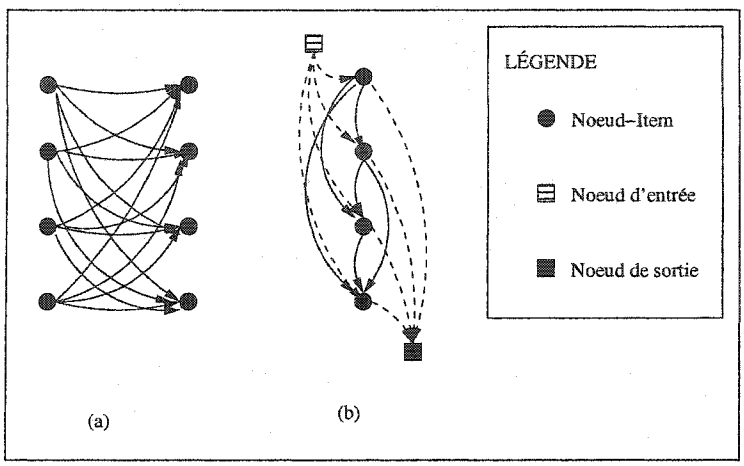

Figure 2.1 – Connexions entre les exemplaires d'un groupe

groupe est relié à chaque exemplaire d'item du groupe par un arc qui représente le choix de chargement de l'exemplaire. Chaque exemplaire d'item d'un groupe est relié au nœud de sortie par un arc qui représente la fin de chargement d'un exemplaire du groupe. Les connexions entre les exemplaires de chaque groupe sont plus restrictives et permettent uniquement les liens entre des exemplaires où le rang du premier est *inférieur* à celui du second. Les connexions entre deux séries successives sont possibles si les groupes en relation sont *différents*. Ces connexions se font entre le nœud de sortie du premier groupe et le nœud d'entrée du second. La Figure 2.2 montre les connexions avant et après les traitements entre les exemplaires de deux groupes différents situés sur deux séries consécutives. Les connexions originales (Figure 2.2 (a)) sont modifiées par l'ajout de nœuds intermédiaires (Figure 2.2 (b)).

Pour distinguer les séries sur le quai des séries sur la rampe, un autre nœud intermédiaire est inséré entre les deux aires dans le réseau. Ce nœud s'interprète comme la fin de chargement sur le quai et/ou le commencement du chargement sur la rampe. Tous les nœuds de sortie de chaque groupe du quai sont reliés par un arc à ce nœud. De même, un arc relie ce nœud à chaque nœud d'entrée de chaque groupe sur la rampe.

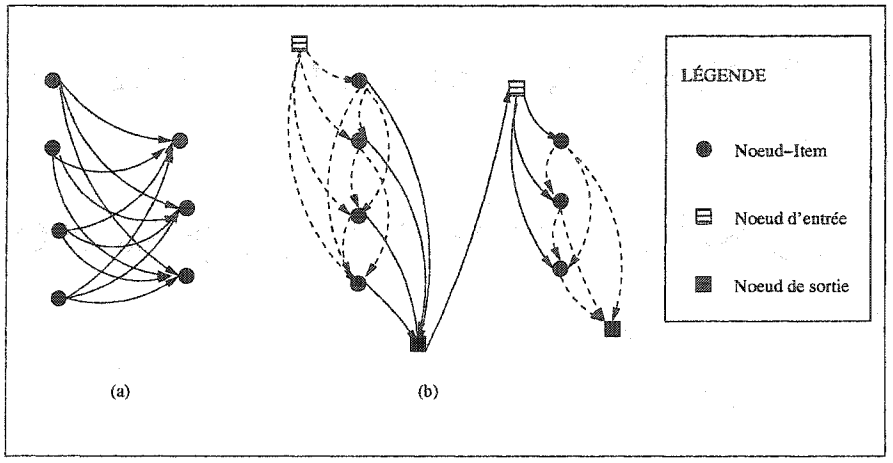

Figure 2.2 – Connexions entre les exemplaires de deux groupes différents

Tous les arcs issus du nœud *source* et allant au nœud d'entrée de chaque sous-groupe de la première série ont un coût de 1. Tous les autres arcs ont un coût de 0.

Ces modifications effectuées sur le réseau permettent de satisfaire les contraintes ajoutées, en l'occurrence, l'interdiction de faire se succéder le même exemplaire d'item sur le quai. Le nombre de séries pouvant suffire pour la construction du réseau se calcule maintenant comme le nombre maximal d'items *différents* placés sur le quai en alternant les deux items les moins contraignants en longueur ou en poids. Dans l'exemple numérique, les deux items les plus courts s'alternent au plus trois fois sur le quai et un seul item peut être chargé sur la rampe. A partir de ces modifications, le nombre d'arcs de connexion entre les exemplaires d'un item w se trouve réduit de q_w^2 à $\frac{(q_w-1)(q_w-2)}{2}$. Le nombre d'arcs reliant deux items de types différents w et w' passe de $q_w \times q_{w'}$ à $q_w + 1 + q_{w'}$. Ces gains sont énormes lorsqu'on aborde des problèmes de taille réaliste. Par exemple, si on a dix types d'items, chacun en dix exemplaires, reliés entre eux sur deux séries consécutives, on aura $100 * 100 = 10\,000$ arcs si aucune amélioration n'est effectuée sur le réseau. Si les arcs horizontaux sont enlevés, il reste $100 * 99 = 9\,900$ arcs. Si, de plus, les arcs verticaux dirigés du bas vers le haut sont enlevés, il reste $(99 + 98 + ... + 90) * 10 = 9\,450$ arcs. Enfin, si des

nœuds intermédiaires sont ajoutés et si nous retenons seulement les arcs où le rang du nœud de départ est *inférieur* à celui du nœud d'arrivée dans un groupe, on aura $(10 + 45 + 10) * 10 + 90 = 740$ arcs. En outre, comme l'algorithme de génération de colonnes est appelé plusieurs fois lors de la résolution du problème, nous avons intérêt à ce que le réseau de chargement soit réduit en termes de nœuds et d'arcs utilisés, sans pour autant omettre de générer des chargements réalisables, et ce, jusqu'à un certain niveau de coût de faisabilité en termes de temps de calcul ou de mémoire CPU.

Nous ne perdons pas de solutions potentielles avec le nouveau réseau. En effet, le second réseau peut générer, à l'instar du premier, tous les chargements comportant uniquement des items différents. Les chargements comportant plusieurs fois le même exemplaire mais alterné par un ou plusieurs items différents, générés par le premier réseau, sont aussi générés par le second. Donc, toute solution réalisable issue du premier réseau peut être retrouvée par le second, à une permutation près, sur le rang des exemplaires.

Avec le nouveau réseau, la relaxation linéaire du problème (tableau 2.3) devient :

Tableau 2.3 – Deuxième relaxation linéaire

min	y_6	$+ y_9$	$+ y_{10}$	$+ y_{11}$	$+ y_{12}$	$+ y_{13}$	$+ y_{14}$		
		y_9				$+ y_{13}$		$= 1$	(item1_1)
				$+ y_{11}$			$+ y_{14}$	$= 1$	(item2_1)
	y_6			$+ y_{11}$				$= 1$	(item2_2)
	y_6						$+ y_{14}$	$= 1$	(item2_3)
	$y_6{}^*$		$+ 2\ y_{10}{}^*$					$= 1$	(item3_1)
					$+ 2\ y_{12}{}^*$	$+ y_{13}{}^*$		$= 1$	(item3_2)
		y_9	$+ y_{10}$		$+ y_{12}$			$= 1$	(item4_1)
			$+ y_{10}$		$+ y_{12}$	$+ y_{13}$		$= 1$	(item4_2)
	$y_6,$	$y_9,$	$y_{10},$	$y_{11},$	$y_{12},$	$y_{13},$	y_{14}	≥ 0	

Une solution optimale est : $y_6 = \frac{2}{4}, y_9 = \frac{2}{4}, y_{10} = \frac{1}{4}, y_{11} = \frac{2}{4}, y_{12} = \frac{1}{4}, y_{13} = \frac{2}{4}, y_{14} = \frac{2}{4}$ et $y_i = 0$ pour tous les autres i. La valeur de la fonction objectif est de $\frac{12}{4} = 3$.

Remarquons que, dans le second réseau, tous les exemplaires succédant un même item dans un chargement apparaissent en ordre croissant (Voir Figure 2.1 (b)) et que le réseau peut générer la succession du même exemplaire d'item sur le quai et sur la rampe, dans un chargement. En effet, lorsqu'un exemplaire d'un item peut se mettre sur la rampe, le groupe apparaît à chaque série du quai et à l'unique série sur la rampe. Ces groupes sont reliés par un arc passant par le nœud *quai-rampe*. Bien que le fait d'ajouter un nœud intermédiaire entre les deux aires de placement réduise le nombre d'arcs du second réseau, la satisfaction de la contrainte imposée n'est plus garantie lorsqu'on passe du quai vers la rampe. Par exemple, dans les plans y_{10} ou y_{12} générés dans la solution précédente, l'item pouvant être mis sur la rampe apparaît deux fois de suite et en succession. Notons que cette difficulté est résolue si nous nous passons du nœud intermédiaire, c'est-à-dire en procédant comme dans le premier réseau et en utilisant beaucoup d'arcs. Nous décidons de laisser à notre méthode de branchement le soin d'éliminer ces chargements.

Si nous ajoutons la contrainte suivante : *pour chaque item de même type, on ne permet pas de charger un exemplaire après l'autre sauf s'ils sont consécutifs dans leur ordonnancement*, le chargement associé à la variable y_{14} n'est plus accepté. Cette contrainte est satisfaite dans le réseau si nous ne considérons uniquement que les connexions entre les nœuds de *rang successif* au lieu des connexions entre des exemplaires où le rang du premier est *inférieur* à celui du second. Avec un réseau de ce type, le nombre d'arcs entre deux séries de l'exemple à dix types d'items précédent est rendu à $(10 + 9 + 10) * 10 + 90 = 380$. En utilisant le même raisonnement que précédemment, nous ne perdons pas non plus de solutions potentielles en nous restreignant à ces connexions. La relaxation linéaire du problème (tableau 2.4) devient :

Une solution optimale est : $y_5 = \frac{1}{2}, y_6 = \frac{2}{2}, y_9 = \frac{1}{2}, y_{12} = \frac{1}{2}, y_{15} = \frac{2}{2}$ et $y_i = 0$ pour tous les autres i. La valeur de la fonction objectif est de $\frac{7}{2} = 3,5$.

Tableau 2.4 – Troisième relaxation linéaire

min	y_5	$+ y_6$	$+ y_9$	$+ y_{12}$	$+ y_{15}$		
	y_5		$+ y_9$			$= 1$	(item1_1)
					$+ y_{15}$	$= 1$	(item2_1)
		$+ y_6$				$= 1$	(item2_2)
		$+ y_6$				$= 1$	(item2_3)
		$+ y_6{}^*$				$= 1$	(item3_1)
				$+ 2\ y_{12}{}^*$		$= 1$	(item3_2)
			$+ y_9$	$+ y_{12}$		$= 1$	(item4_1)
	y_5			$+ y_{12}$		$= 1$	(item4_2)
	$y_5,$	$y_6,$	$y_9,$	$y_{12},$	y_{15}	≥ 0	

Enfin la formulation en nombres entiers (tableau 2.5) du problème nous donne les valeurs des variables suivantes :

Tableau 2.5 – Formulation en nombres entiers de l'exemple

min	y_9	$+ y_{15}$	$+ y_{16}$	$+ y_{17}$			
	y_9					$= 1$	(item1_1)
		$+ y_{15}$				$= 1$	(item2_1)
				$+ y_{17}$		$= 1$	(item2_2)
				$+ y_{17}$		$= 1$	(item2_3)
			$+ y_{16}$			$= 1$	(item3_1)
			$+ y_{16}$			$= 1$	(item3_2)
	y_9					$= 1$	(item4_1)
			$+ y_{16}$			$= 1$	(item4_2)
	$y_9,$	$y_{15},$	$y_{16},$	y_{17}	≥ 0		

$y_9 = y_{15} = y_{16} = y_{17} = 1$ et $y_i = 0$ pour tous les autres i. La valeur de la fonction objectif est 4. Notons qu'il n'existe pas de solution à trois avions et l'intérêt d'avoir une meilleure borne inférieure s'avère essentiel car cela constitue une première étape dans un arbre d'énumération.

D'après l'étude de l'exemple précédent, nous avons la proposition suivante.

Proposition 1 *Dans le cas particulier où le transport des items se fait entre une seule ville de collecte et une seule ville de destination, ALP ne possède pas la propriété*

de l'arrondissement supérieur ou IRUP (integer round-up property), c'est-à-dire que la solution optimale en nombres entiers du problème n'est pas toujours la solution de la relaxation arrondie à l'entier suivant par rapport à la formulation se basant sur le réseau "naïf".

2.3 Le réseau de chargement

Le réseau de chargement décrit ci-dessous découle de l'étude de l'exemple précédent. Pour faciliter la compréhension du réseau et pour ne pas alourdir la description du réseau, nous supposons, dans cette section, que les items à transporter ont la même priorité d'envoi. Bien qu'une instance où plus d'une priorité d'envoi ait été étudiée dans la section précédente, le cas général sera présenté dans la section 2.5.

Notons par N, indexé par w, l'ensemble de tous les items à charger. Soit K l'ensemble des avions et soit $G^k = (V^k, A^k)$ le réseau de chargement associé à l'avion $k \in K$ illustré par la Figure 2.3. Si O^k et D^k sont deux nœuds particuliers du réseau appelés respectivement la *source* ou la base de sortie de l'avion k et le *puits* ou sa base de retour, alors dénotons par $V^k = N^k \bigcup I^k \bigcup \{O^k, D^k\}$ l'ensemble des nœuds. N^k désigne l'ensemble restreint des nœuds associés aux items pouvant être chargés dans l'avion k ($N = \bigcup_{k \in K} N^k$). I^k désigne l'ensemble des nœuds intermédiaires qui ne sont pas associés à des items, ni à O^k, ni à D^k mais qui sont conçus pour interpréter facilement le réseau et/ou pour réduire le nombre d'arcs requis. Soit A^k l'ensemble des arcs du réseau qui décrivent les connexions entre les nœuds. Nous supposons qu'aucun arc n'aboutit au nœud O^k et qu'aucun arc ne prend origine au nœud D^k. Tous les chemins commencent par le nœud O^k et se terminent au nœud D^k.

Nous avons six types de nœuds : la *source*, le *puits*, le *quai-rampe*, les nœuds associés aux exemplaires d'items ou nœud-items, les nœuds d'entrée et de sortie localisés au sein d'un sous-groupe formé par les exemplaires d'un même type d'item.

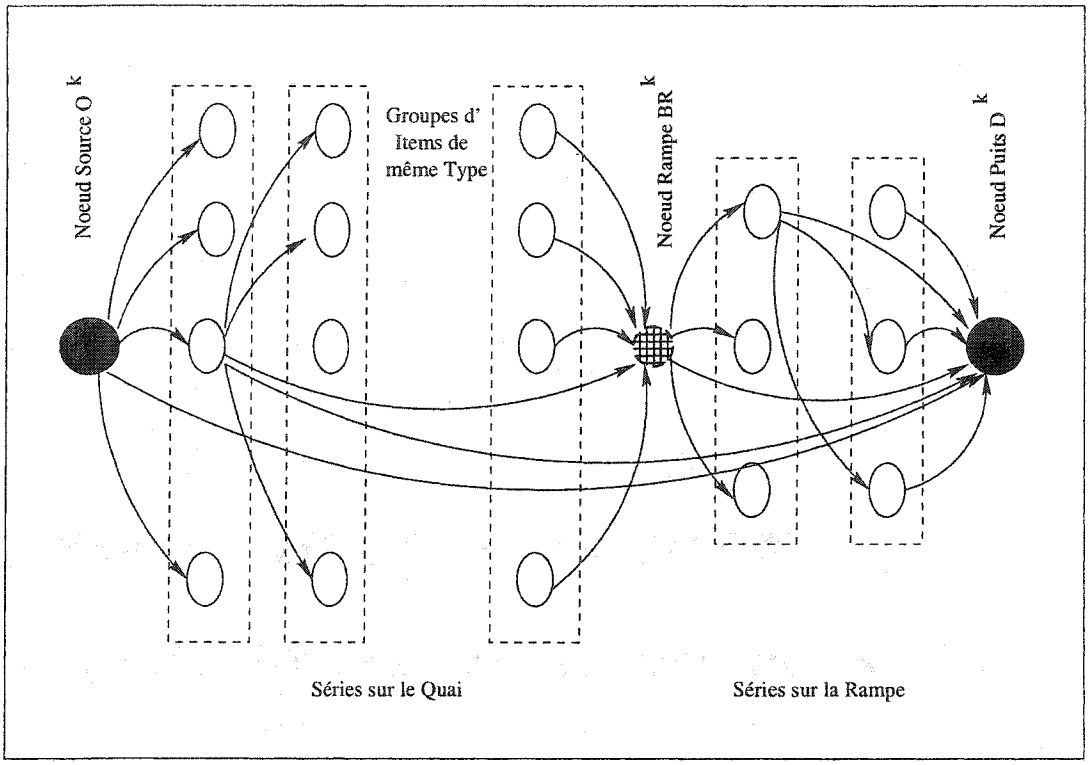

Figure 2.3 – Le réseau de chargement

Le premier nœud O^k est la *source* indiquant le commencement du chargement. Le chargement est fini au nœud *puits* D^k. Puisque l'aire de chargement est divisée en deux parties, en l'occurrence, le quai et la rampe, ayant chacune ses propres contraintes structurelles, le nœud intermédiaire *quai-rampe* BR^k sépare le chargement en deux : les items qui sont chargés sur le quai et ceux sur la rampe. Précisons, pour faciliter la compréhension, qu'on emploiera indifféremment dans le texte les termes *item à charger* ou le nœud correspondant. Un nœud $I^k_{w\, z_w\, s}$ est défini pour chaque exemplaire $z_w, z_w = 1, ..., q_w$ de l'item $w \in N^k$ lorsque celui-ci peut être mis à la position s d'un chargement réalisable. Des copies de ce nœud peuvent exister dans toutes les positions, à la fois sur le quai ou sur la rampe, si l'exemplaire est susceptible d'être mis à ces positions. Notons que les nœuds représentant le même exemplaire contribuent, de ce fait, à la même tâche. Les nœuds sont disposés en sé-

ries : une première série contient tous les nœuds associés aux exemplaires d'items qui sont susceptibles d'être placés à l'avant de l'avion, une seconde série contient ceux associés aux items occupant le seconde position, etc. Comme nous avons supposé que les items sont assez grands, le nombre de ces séries est réduit et est égal au nombre maximal d'items qu'on peut charger dans l'avion, lequel nombre est connu une fois que les items à charger sont recensés. Soit S l'ensemble de toutes les séries, et s un élément de cet ensemble. Pour chaque série, les nœuds sont mis ensemble selon l'item associé dans un groupe (Voir Figure 2.4). Un nœud intermédiaire B^k_{ws} indique l'entrée dans le groupe w de la série s pour la commodité k, et un autre nœud intermédiaire E^k_{ws} pour la sortie du groupe.

L'ensemble d'arcs A^k est composé de six types d'arcs :

1. Les arcs incidents à un type d'items (Voir Figure 2.3) :
   - Les arcs $(O^k, B^k_{w1}), w = 1, ..., |N^k|$ sont définis pour la sélection de l'item placé à la première position. Ils décrivent toutes les possibilités de choix du premier item pouvant être mis sur le quai. Ces arcs relient la *source* avec la première série sur le quai.
   - Les arcs $(E^k_{ws}, B^k_{w'\,s+1}), w, w' = 1, ..., |N^k|, w \neq w'$, sont définis pour la sélection de l'item w' de la série $s+1$ immédiatement après le chargement d'un exemplaire de l'item w de la série s. Ils permettent de changer d'items et ils relient deux groupes de deux séries consécutives, sur le quai et sur la rampe. Nous imposons la condition $w \neq w'$ (le cas $w = w'$ est considéré dans l'étude du cas du troisième ensemble d'arcs).
   - Les arcs $(BR^k, B^k_{w\,s}), w = 1, ..., |N^k|$ sont définis pour la sélection du premier item chargé sur la rampe. Ils décrivent toutes les possibilités de choix du premier item pouvant être mis sur la rampe. Ces arcs relient le *quai-rampe* avec la première série sur la rampe.
2. Les arcs $(B^k_{w\,s}, I^k_{w\,z_w\,s}), z_w = 1, ..., q_w$, incidents à un exemplaire de l'item w de la série s sont définis pour la sélection de celui qui sera chargé en premier dans

la série s. Nous les trouvons au sein d'un même groupe, sur le quai ou sur la rampe. (Voir Figure 2.4)

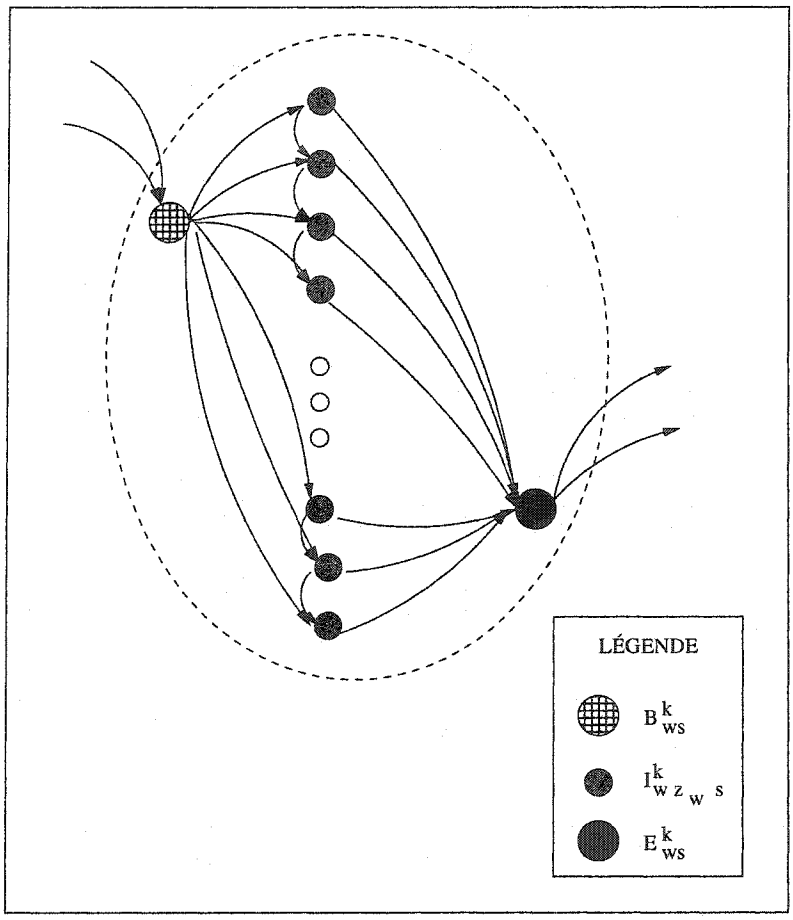

Figure 2.4 – Connexions entre les exemplaires d'un item

3. Les arcs $(I^k_{w\,z_w\,s}, I^k_{w\,z_w+1\,s})$, $z_w = 1, ..., q_w - 1$ reliant deux exemplaires de l'item w de la série s sont définis lorsque les deux peuvent être chargés successivement. Ce sont des arcs de suivi et nous les trouvons au milieu d'un groupe sur le quai ou sur la rampe. (Voir Figure 2.4)
4. Les arcs $(I^k_{w\,z_w\,s}, E^k_{ws})$, $z_w = 1, ..., q_w$, incidents au nœud de sortie du groupe d'items w de la série s sont définis pour indiquer la fin du chargement de l'item

dans la série correspondante. Nous les trouvons à l'intérieur d'un même groupe, sur le quai ou sur la rampe. (Voir Figure 2.4)

5. Les arcs $(E^k_{w\,s}, BR^k)$ reliant le groupe d'items w de la série s avec le nœud *quai-rampe* sont définis pour indiquer la fin du chargement sur le quai, pour chaque groupe et chaque série ; et les arcs $(E^k_{w\,s}, D^k)$, reliant le groupe d'items w de la série s avec le nœud *puits* sont définis pour indiquer la fin du chargement, pour chaque groupe et chaque série. (Voir Figure 2.3)
6. L'arc (O^k, D^k) reliant la *source* et le *puits* est défini lorsque l'avion k reste vide. (Voir Figure 2.3)

Nous supposons qu'un coût c^k_{ij} est associé à l'arc $(i,j) \in A^k$. En général, ce coût peut être la distance parcourue par l'avion ou l'essence consommée par l'avion ou d'autres mesures de coût, donc il tient compte implicitement, entre autres, de la nature de la cargaison et/ou du temps passé pour charger les items lorsque l'arc $(i,j) \in A^k$ est parcouru. Dans ce chapitre, comme nous minimisons le nombre d'avions utilisés, un coût de 1 est associé à tous les arcs $(O^k, B^k_{w1}), w = 1, ..., |N^k|$ et un coût de 0 pour tous les autres arcs. Ce coût non nul est chargé pour chaque unité de flot qui passe par l'arc. Cela correspond à la création d'un nouveau plan de chargement.

La Figure 2.5 montre un réseau de chargement où les arcs en gras décrivent un chargement réalisable. Les copies de l'item 2 sont les seuls items qui peuvent être mis sur la rampe. Nous avons deux séries sur le quai car nous ne pouvons pas y mettre plus de deux séquences d'items différents. Par ailleurs, une seule série est requise sur la rampe. Le chemin plus gras indique un chargement où les deux items du troisième type sont chargés sur le quai et le premier item du deuxième type est chargé sur la rampe.

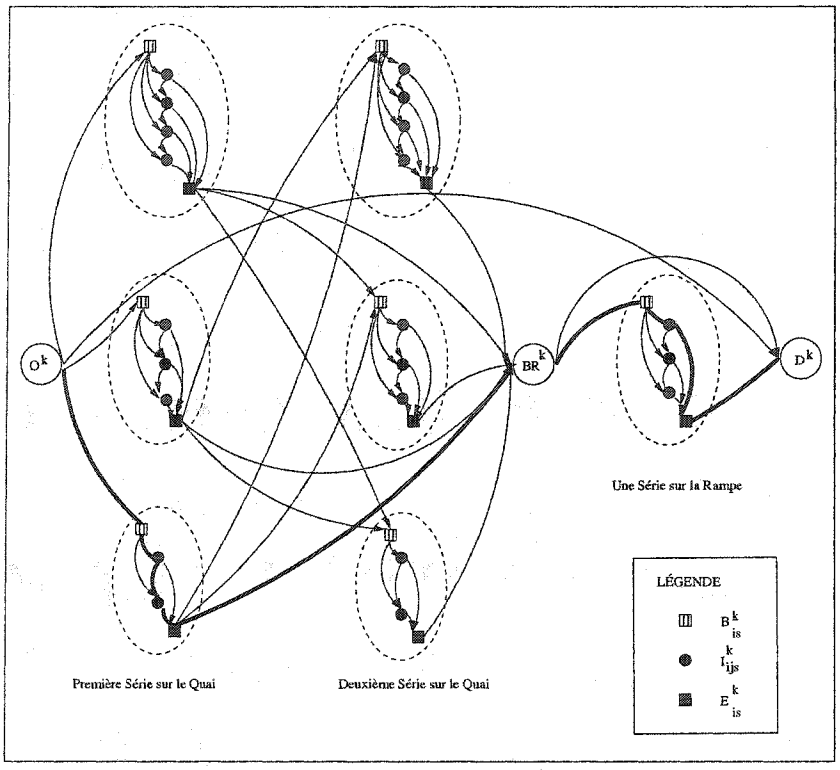

Figure 2.5 – Exemple de réseau de chargement

2.4 Traitement des cas spéciaux

Si des passagers sont transportés, ils doivent être assis à l'avant de l'avion. Un nombre suffisant de sièges mobiles ont été conçus à cet effet et ces sièges sont placés par rangée. Par conséquent, nous pouvons considérer plusieurs rangées comme un item spécial. Ainsi, un groupe de nœuds est ajouté *avant* l'ensemble des groupes de la première série. Le nœud *source* est en connexion, soit avec ce groupe, soit avec des groupes de la première série. En effet, cela évite que les passagers soient placés derrière un item quelconque, et cela n'impose pas que chaque avion doive charger des passagers. Dans ce groupe, nous associons des nœuds à une ou plusieurs rangées de sièges et deux autres nœuds intermédiaires sont respectivement associés à l'entrée et à la sortie du groupe.

Quand des priorités d'envoi sont imposées, nous construisons un réseau de chargement pour chaque groupe d'items de même priorité ou de priorités successives. Le cas de trois priorités a déjà fait l'objet d'une discussion dans l'exemple étudié, et nous avons considéré deux sous-réseaux, respectivement pour les items de priorités 1 et 2, et ceux de priorités 2 et 3. Le cas général peut en être déduit facilement. Une copie d'un exemplaire d'item peut se retrouver alors dans plusieurs positions des aires de placement et possiblement dans deux sous-réseaux différents.

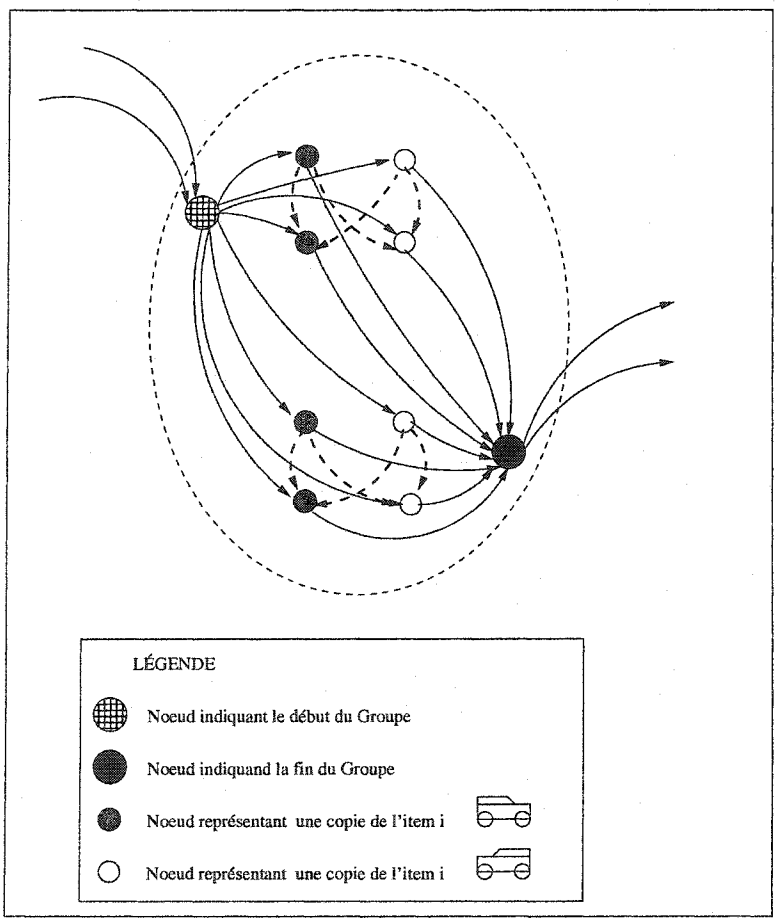

Figure 2.6 – Groupe d'items dont le CG n'est pas en leur milieu

Si le CG d'un item n'est pas en son milieu, comme le cas des hélicoptères qui sont démontés avant d'être embarqués, le modèle peut être aussi adapté. Pour chaque item

vérifiant ces conditions, nous pouvons trouver des exemplaires dans les différentes positions possibles, sur le quai ou sur la rampe. Pour chaque position, pour chacun de ces items et pour chacun de ses exemplaires, deux nœuds sont associés : un pour représenter l'exemplaire chargé dans le même sens que la tête de l'avion, et un autre pour représenter le chargement dans le sens contraire (Voir Figure 2.6). Deux autres nœuds intermédiaires sont ajoutés respectivement pour l'entrée et la sortie du groupe. Chacun des deux nœuds représentant un exemplaire est relié, soit au nœud d'entrée, soit au nœud de sortie. Aucun arc ne relie entre eux les deux nœuds représentant chaque exemplaire et ce, pour exprimer le choix exclusif de celui-ci à cet emplacement dans le réseau. Par contre, lorsqu'il est possible de charger successivement deux exemplaires, nous pouvons les charger soit dans le même sens, soit le premier tourné vers la tête de l'avion et le second dans le sens opposé ou vice versa. Les arcs en diagonal (traits discontinus) de la Figure 2.6 illustrent ces possibilités.

Enfin, si des contraintes de préséance se présentent, tel le cas des remorques qui doivent être tirées lors du chargement ou poussées lors du déchargement, le modèle peut être adapté. Nous créons pour ces items un groupe dans chaque série du réseau. Comme il n'est pas possible de faire succèder deux items de ces types, les arcs de suivi sont alors enlevés dans le groupe. De plus, ces remorques ne peuvent être tirées que par un ensemble d'items défini au préalable. De ce fait, les arcs incidents à ce type d'items sont réduits. Dans chaque série du réseau, l'arc entrant au groupe associé à l'item remorqué doit être incident à un groupe correspondant à un remorqueur. L'arc sortant au groupe associé à l'item remorqué doit être incident soit à un groupe quelconque et différent, soit au nœud $quai - rampe$, soit au nœud $puits$.

2.5 Formulation mathématique

Dans ce qui suit, nous cherchons un chemin reliant l'origine O^k à la destination D^k, pour tout $k \in K$ tel que chaque item à transporter ait un de ses nœuds $I^k_{w\, z_w\, s}$ dans

un et un seul chemin. Chacun de ces chemins devra répondre à certaines caractéristiques correspondant aux contraintes structurelles des chargements. On modélise ces contraintes au moyen de ressources. Par exemple, on peut considérer le plancher comme 492 pouces de ressource. Si on y place un item de 75 pouces, il ne restera que 417 pouces de ressource. Nous aurons donc des variables de flot pour décrire le chemin dans le réseau et des variables de ressources pour comptabiliser la consommation des ressources dans les chemins sélectionnés. Les variables représentant le flot dans le réseau G^k sont notées par le vecteur $\mathbf{X}^k = (X_{ij}^k | (i,j) \in A^k)$, où X_{ij}^k prend la valeur 1 si l'arc (i,j) est parcouru c'est-à-dire si le nœud j est visité immédiatement après le nœud i dans le chemin correspondant au chargement de l'avion k, et la valeur 0 sinon. Notons par R^k, indicé par r, l'ensemble des ressources utilisées dans G^k. Les variables représentant le niveau des ressources à chaque nœud de V^k sont notées par le vecteur $\mathbf{T}^k = (T_i^{kr} | i \in V^k, r \in R^k)$, où T_i^{kr} prend la valeur cumulée de la ressource r sur un chemin parcouru dans le réseau depuis le nœud O^k jusqu'au nœud i, et la valeur 0 si le nœud i n'est pas visité. Notons par $\mathbf{T}_i^k = (T_i^{kr} | r \in R^k)$ le vecteur de toutes les ressources au nœud $i \in V^k$. Notons par f_{ij}^{kr} la fonction qui prolonge le calcul de la consommation de la ressource r dans G^k du nœud i au nœud j. $f_{ij}^{kr}(\mathbf{T}_i^k)$ représente la quantité de ressource $r \in R^k$ consommée dans un chemin reliant les nœuds O^k et j en passant par l'arc $(i,j) \in A^k, k \in K$.

Notons par $a_{w,ij}^k$ le coefficient binaire de la variable de flot X_{ij}^k qui prend la valeur 1, si l'arc $(i,j) \in A^k$ est parcouru dans le réseau et un exemplaire de l'item w est chargé par cette action au nœud j, et 0 sinon. Notons enfin, par q_w le nombre d'exemplaires de l'item du type w, par a_i^{kr} et b_i^{kr} respectivement la borne inférieure et la borne supérieure de l'intervalle d'admissibilité pour le niveau de ressource T_i^{kr} au nœud $i \in N^k$ pour chaque ressource r et pour tout $k \in K$.

La formulation de ALP est :

$$z_{IP} = \min \sum_{k \in K} \sum_{w \in N} X^k_{O^k B_{w1}} \tag{2.1}$$

sujet aux contraintes :

$$\sum_{k \in K} \sum_{s \in S} \sum_{(i,j) \in A^k : j = I^k_{w\, z_w\, s}} a^k_{w,ij} X^k_{ij} = 1, \forall w \in N, \forall z_w = 1, ..., q_w \tag{2.2}$$

$$\sum_{j:(O^k,j) \in A^k} X^k_{O^k j} = \sum_{j:(j,D^k) \in A^k} X^k_{jD^k} = 1, \forall k \in K \tag{2.3}$$

$$\sum_{j:(i,j) \in A^k} X^k_{ij} - \sum_{j:(j,i) \in A^k} X^k_{ji} = 0, \forall k \in K, \forall i \in V^k \tag{2.4}$$

$$X^k_{ij} \in \{0,1\}, \forall k \in K, \forall (i,j) \in A^k \tag{2.5}$$

$$X^k_{ij}(f^{kr}_{ij}(\mathrm{T}^k_i) - T^{kr}_j) = 0, \forall k \in K, \forall r \in R^k, \forall (i,j) \in A^k \tag{2.6}$$

$$a^{kr}_i \leq T^{kr}_i \leq b^{kr}_i, \forall k \in K, \forall r \in R^k, \forall i \in \{O^k, D^k\} \tag{2.7}$$

$$a^{kr}_i (\sum_{j:(i,j) \in A^k} X^k_{ij}) \leq T^{kr}_i \leq b^{kr}_i (\sum_{j:(i,j) \in A^k} X^k_{ij}), \forall k \in K, \forall r \in R^k, \forall i \in V^k. \tag{2.8}$$

La fonction objectif (2.1) compte le nombre d'avions-cargos requis pour transporter tous les items. Les contraintes (2.2) imposent que le $z^{ième}_w$ exemplaire de l'item w soit chargé exactement une fois. Le $z^{ième}_w$ exemplaire de l'item w est représenté par plusieurs nœuds dans le réseau, selon sa position dans l'ordre de chargement. C'est pourquoi on se doit de sommer sur tous les arcs dont la destination est un de ces nœuds. Ce sont des contraintes globales coordonnant les avions entre eux, appelées *multiple path linking constraints* dans Desaulniers *et al.* (1998). Les contraintes (2.3)- (2.8) regroupent les variables par avion. Ce sont des contraintes locales et elles sont dénommées *single path constraints* dans Desaulniers *et al.* (1998). Les contraintes (2.3)-(2.5) définissent la structure d'un chargement ou chemin unique pour chaque avion k dans le réseau G^k. Ainsi, les contraintes (2.3) et (2.4) sont les contraintes de chemin dans un réseau ; elles indiquent qu'une seule unité de flot doit être envoyée à partir

du nœud *source* O^k vers le nœud *puits* D^k pour chaque réseau G^k et que les flots doivent être conservés à chaque nœud. Une des contraintes (2.3) est redondante et pourrait être enlevée. Les contraintes (2.5) imposent que les variables de flots doivent être binaires. Les contraintes (2.6) assurent la compatibilité entre les variables de flot et les variables de ressource, autrement dit, la ressource consommée au nœud j doit être égale à celle au nœud i à laquelle on a fait une extension lors du parcours de l'arc (i, j). Les contraintes (2.7) et (2.8) sont les restrictions portant sur chaque variable de ressource T_i^{kr}, $r \in R^k$, soit son intervalle d'admissibilité si le nœud i est visité dans G^k, et dans le cas contraire cela implique que cette variable est nulle.

L'arc (O^k, D^k) est inclus dans le réseau pour modéliser l'utilisation des avions chargés à vide. Le modèle permet alors de fixer une borne sur le nombre d'avions utilisés. Si cette borne est suffisamment grande, le nombre d'avions est dit libre. Une borne inférieure sur le nombre d'avions utilisés peut être aussi prise en compte. Ainsi, $c^k_{O^kD^k} = 0$ tiendra compte du fait que nous avons une flotte illimitée d'avions. Ceux qui ne sont pas utilisés dans la solution optimale vont utiliser cet arc. Si on veut chercher une solution où le nombre d'avions sera $|K|$, alors il suffit de fixer $c^k_{O^kD^k}$ à une très grande valeur et dès lors, l'arc ne sera jamais parcouru dans une solution optimale. De même, si on veut que le nombre d'avions soit libre et qu'il faille tenir compte d'un coût fixe c^k_{fixe} pour utiliser un avion k, alors on posera $c^k_{O^kD^k} = -c^k_{fixe}, \forall k \in K$. Si le coût fixe c^k_{fixe} est assez grand, le modèle minimise dans un premier temps le nombre d'avions utilisés et par la suite les autres coûts, entre autres, les coûts reliés à la distance si plusieurs villes de collecte sont prises en compte dans l'optimisation.

Nous revenons aux équations (2.6) qui complètent la description de ALP au moyen de variables de ressource. Ces variables de ressource sont importantes car elles modélisent les contraintes locales de chargement. En effet, elles sont cumulées sur les arcs d'un chemin et sont assujetties à des contraintes qui assurent la faisabilité des chargements, lesquels sont considérés comme des suites d'arcs. Les restrictions sont imposées sur

chaque nœud du réseau et sont vérifiées et mises à jour par des transformations non linéaires qui permettent de prolonger ou d'étendre les informations d'un nœud à son successeur. Nous allons spécifier ces variables de ressource avec leurs bornes respectives et les fonctions de prolongation correspondantes.

2.5.1 La ressource longueur totale

La première variable de ressource T_i^{k1} correspond à la longueur totale des items déjà chargés dans l'avion k, une fois mis bout à bout, à partir du nœud *source* jusqu'au nœud i. La fonction de prolongation f_{ij}^{k1} s'écrit comme suit :

$$f_{ij}^{k1}(\mathrm{T}_i^k) = \begin{cases} T_i^{k1} + l_j & \text{si } j \text{ correspond à un item et } i \neq BR^k, \\ L_B & \text{si } i = BR^k, \\ T_i^{k1} & \text{sinon.} \end{cases} \tag{2.9}$$

Dans cette équation, l_j est la longueur de l'exemplaire de l'item w associé à ce nœud j. L_B désigne la longueur du quai et L désigne la longueur de l'avion. Pour tous les nœuds i correspondants à des positions situées sur le quai, la valeur de T_i^{k1} doit toujours être plus petite ou égale à la longueur du quai et nous avons : $0 \leq T_i^{k1} \leq L_B$. Ainsi, dans la contrainte (2.7), en posant b_i^{kr} égal à la longueur du quai pour chacun de ces nœuds i, on évite qu'un item chevauche le quai et la rampe. Elle offre, en outre, la possibilité de charger directement la rampe sans épuiser l'espace restant sur le quai. Nous retrouvons une version de cette alternative dans les problèmes de tournées de véhicules avec fenêtres de temps lorsque le véhicule arrive trop tôt pour effectuer sa tâche, la deuxième partie de la définition (2.9) permet ainsi des attentes. Pour les nœuds i correspondants à des positions situées sur la rampe, T_i^{k1} doit toujours être plus petite ou égale à la longueur de l'avion L. La contrainte (2.7) s'écrit : $L_B \leq T_i^{k1} \leq L$ pour ces nœuds. La fonction de prolongation nous informe, entre autres, que lorsque nous commençons à charger sur la rampe, la longueur totale des items déjà chargés est considérée comme étant la longueur du quai.

2.5.2 La ressource poids total

La seconde variable de ressource T_i^{k2} correspond au poids total des items déjà chargés à compter du nœud *source* jusqu'au nœud i de l'avion k. La fonction de prolongation est similaire à l'équation (2.9), sauf que la longueur l_j est remplacée par le poids m_j et il n'y a pas de mise à jour au nœud *quai-rampe*. La fonction de prolongation correspondante est :

$$f_{ij}^{k2}(\mathbf{T}_i^k) = \begin{cases} T_i^{k2} + m_j & \text{si } j \text{ correspond à un item,} \\ T_i^{k2} & \text{sinon.} \end{cases} \tag{2.10}$$

La valeur de T_j^{k2} doit être inférieure ou égale à la limite maximale permise de poids du chargement de l'avion k, notée M. La contrainte (2.7) s'écrit alors : $0 \leq T_i^{k2} \leq M$ pour tous les nœuds du réseau.

2.5.3 La ressource poids sur la rampe

Une variable spécifique de ressource T_i^{k3} est nécessaire pour limiter le poids des items chargés sur la rampe. La fonction de prolongation $f_{ij}^{k3}(\mathbf{T}_i^k)$ est similaire à celle de l'équation (2.10) excepté que l'incrémentation de la variable est effectuée uniquement pour les items sur la rampe. La valeur de T_j^{k3} doit être plus petite ou égale à la limite maximale requise de poids alloué sur la rampe, notée M_R. La contrainte (2.7) s'écrit alors : $0 \leq T_i^{k3} \leq M_R$ pour tous les nœuds i correspondants à des positions situées sur la rampe.

2.5.4 La ressource centre de gravité

Soit I^k l'ensemble des items déjà chargés dans l'avion k. Soit j le dernier item chargé et i l'avant dernier. La position x_j^k du CG de l'item j, de longueur l_j et de poids m_j,

se calcule comme suit :

$$x_j^k = T_i^{k1} + \alpha_j l_j$$

où T_i^{k1} correspond à la longueur courante totale des items chargés.
Le CG de la cargaison, noté CG^k, de l'avion k est :

$$CG^k = \frac{\sum_{i \in I^k} m_i x_i^k}{\sum_{i \in I^k} m_i}. \tag{2.11}$$

A partir de ces remarques, nous déduisons que le CG d'une cargaison quelconque peut se calculer de manière récurrente selon l'ordre des items contenus dans l'avion. Nous avons besoin de deux variables de ressource pour modéliser le CG : une pour gérer la contrainte de borne supérieure et une autre pour celle de borne inférieure. Notre idée est de calculer deux CG de la même cargaison : la première, lorsque les items sont tassés le plus près possible de la tête de l'avion et la seconde, lorsqu'ils sont tassés le plus près possible de la queue de l'avion. Ainsi, les deux valeurs sont calculées et vérifiées à chaque nœud du réseau par rapport aux bornes associées respectives. La proposition suivante résume notre approche et la figure qui suit illustre la procédure.

Proposition 2 *Une condition nécessaire et suffisante pour assurer la faisabilité de la contrainte du CG de la cargaison est d'assurer que la première valeur du CG doit être plus petite que la limite supérieure fixée pour le CG en tous nœuds et que la seconde doit être plus grande que la limite inférieure fixée pour le CG au dernier nœud puits du réseau.*

Pour la contrainte de borne supérieure, nous suggérons la variable de ressource suivante :

$$f_{ij}^{k4}(\mathrm{T}_i^k) = \begin{cases} \frac{T_i^{k2} T_i^{k4} + m_j (T_i^{k1} + \alpha_j l_j)}{T_i^{k2} + m_j} & \text{si } j \text{ correspond à un item,} \\ T_i^{k4} & \text{sinon.} \end{cases} \tag{2.12}$$

Comme les items sont chargés de la tête de l'avion vers la rampe, la position du CG de la cargaison croît au fur et à mesure qu'on rajoute des items. A chaque nœud, nous devons vérifier si la valeur du CG ne dépasse pas la limite supérieure $\overline{g}$:

$$T_j^{k4} \leq \overline{g}.$$

Si à un nœud du réseau le CG est sous sa limite inférieure $\underline{g}$, il existe deux façons d'augmenter cette valeur : ajouter plus d'items ou pousser tous les items chargés vers le fond de l'avion, si l'espace le permet (Figure 2.7). Dans ce dernier cas, il suffit de vérifier si le CG des items, chargés aussi loin que possible de la tête de l'avion, satisfait la contrainte de borne inférieure.

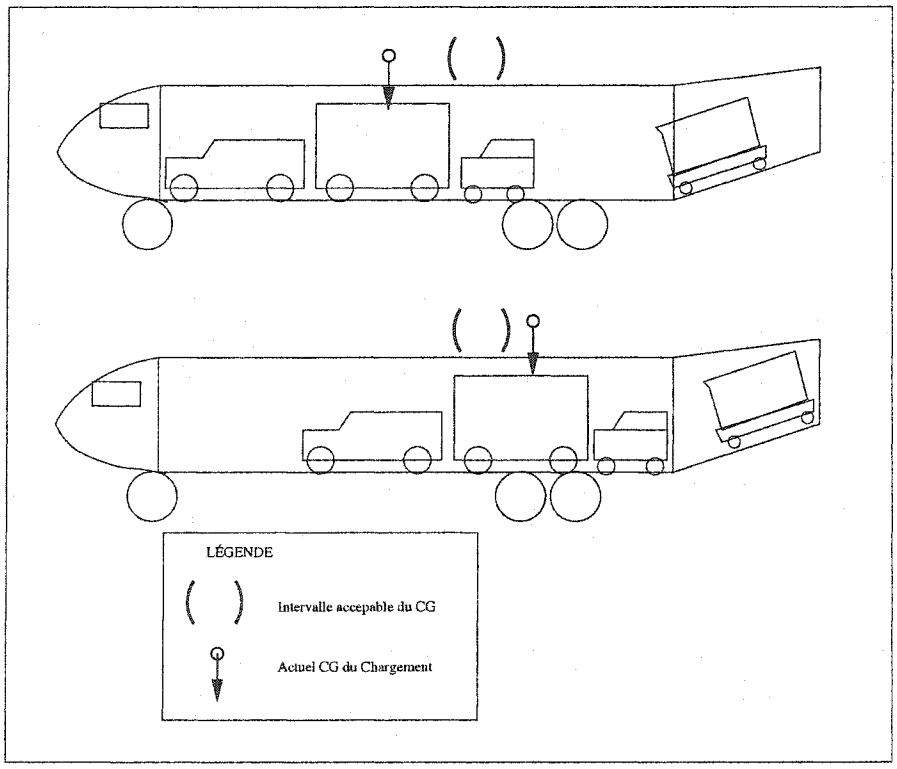

Figure 2.7 – Exemple de chargement réalisable

La nouvelle valeur du CG se calcule comme suit :

$$f_{ij}^{k5}(\mathbf{T}_i^k) = \begin{cases} \frac{T_i^{k2}(T_i^{k5}-l_j)+m_j(L_B-(1-\alpha_j)l_j)}{T_i^{k2}+m_j} & \text{si } j \text{ est un item sur le quai,} \\ \frac{(T_i^{k2}-T_i^{k3})T_i^{k5}+T_i^{k3}(T_i^{k5}-l_j)+m_j(L-(1-\alpha_j)l_j)}{T_i^{k2}+m_j} & \text{si } j \text{ est un item sur la rampe,} \\ T_i^{k5} & \text{sinon.} \end{cases} \tag{2.13}$$

Cette ressource traduit le résultat des déplacements de tous les items chargés vers l'arrière de chaque aire de placement. Pour chaque nœud j du réseau, la valeur de T_j^{k5} est contrainte à être positive, sauf pour le dernier nœud D^k du réseau. A ce nœud D^k, aucun item n'est plus chargé. Donc, la valeur de T_j^{k5} doit être supérieure ou égale à la limite inférieure du CG, c'est-à-dire,

$$\underline{g} \leq T_j^{k5}.$$

2.6 Méthode de résolution

Dans cette section, nous présentons une méthode de *B&B* pour résoudre le modèle (2.1)- (2.8) pour laquelle les bornes inférieures sont obtenues en utilisant l'approche de génération de colonnes. Nous supposons que le nombre d'avions-cargos utilisés est plus grand que un. Sinon, le problème se réduit à un *TSPR* (ou le problème de voyageur de commerce avec contraintes de ressource), et peut être résolu efficacement par la programmation dynamique lorsque les bornes sur les ressources sont assez serrées (Desaulniers *et al.*, 1998). Dans un premier temps, nous précisons le problème maître et le sous-problème. L'algorithme servant à résoudre le sous-problème est décrit par la suite et nous proposons des critères de dominance pour réduire l'espace d'états et faire en sorte que l'algorithme fonctionne plus efficacement. Les règles de branchement sont comme celles de l'approche exacte de Desrochers et Soumis (1989). Enfin, d'autres contraintes sont ajoutées pour améliorer la borne inférieure du *LP* à travers l'arbre de recherche.

2.6.1 Le problème maître et le sous-problème

La formulation (2.1)-(2.8) de *ALP* présente une structure diagonale par blocs (2.3), (2.4), (2.6), (2.7), (2.8), des contraintes de liaisons (2.2) (les items à charger), et une fonction objectif (2.1) qui serait séparable suivant les blocs s'il n'y avait pas les contraintes de liaison. Elle se prête bien à une décomposition de Dantzig-Wolfe. Notre motivation n'est pas d'accélerer la résolution de la relaxation linéaire - au contraire, la relaxation pourrait être plus difficile à résoudre - mais plutôt d'obtenir une meilleure borne inférieure. Il suffit de remarquer que les solutions fractionnaires qui sont des combinaisons convexes des solutions du sous-problème sont aussi réalisables pour la relaxation suite à cette procédure. Notons que les domaines des sous-problèmes ne sont pas convexes et les combinaisons convexes de points extrêmes ne donnent qu'une approximation extérieure de ces ensembles. La décomposition nous permet aussi d'étudier directement le problème à partir des plans de chargement réalisables. A chaque plan de chargement, nous associons une variable du problème maître et la reformulation mettra en cause implicitement un nombre considérable de ces variables. En fait, ces variables ne sont générées par un sous-problème que si besoin est. Ce sous-problème décrit les contraintes entre l'avion et les items à transporter. Son rôle est de fournir le meilleur plan susceptible d'améliorer le nombre d'avions requis. Sa structure est celle d'un problème de plus court chemin avec contraintes de ressource. A partir de cette reformulation, nous pensons aussi éliminer certaines symétries du problème qui peuvent entraver le branchement. Les détails de la procédure de décomposition, tirés de Desaulniers *et al.* (1998), sont décrits en annexe. Nous allons maintenant formuler cette décomposition dans la suite.

Soit Ω l'ensemble de tous les plans réalisables, au sens qu'ils respectent les contraintes de ressource (2.9) à (2.13). Comme le coût de sortie d'un avion est de 1 unité, le coût d'un plan de chargement $e \in \Omega$ est aussi de 1 unité. Soit $\theta_e, e \in \Omega$ la variable

indiquant le nombre de fois où le plan e sera utilisé. θ_e est une variable binaire puisque chaque item doit être transporté en un seul morceau une et une seule fois. Notons par $a_{z_w e}, w \in N, z_w = 1, ..., q_w$ le coefficient binaire indiquant si l'item $z_w, w \in N$ à transporter est présent ($a_{z_w e} = 1$) dans le chargement $e \in \Omega$ ou non ($a_{z_w e} = 0$). Le problème maître peut donc s'écrire :

$$\min \sum_{e \in \Omega} \theta_e \tag{2.14}$$

sujet aux contraintes :

$$\sum_{e \in \Omega} a_{z_w e} \theta_e = 1, \forall w \in N, z_w = 1, ..., q_w \tag{2.15}$$

$$\theta_e \geq 0, entiers, \forall e \in \Omega. \tag{2.16}$$

Le problème maître est écrit en termes de plan de chargement et la fonction objectif minimise le nombre de chargements ou, en d'autres termes, le nombre d'avions requis. Les contraintes (2.15) imposent à chaque type d'item, l'obligation pour chaque exemplaire, d'être chargé dans un et un seul plan de chargement. Et pour finir, nous exigeons que les plans de chargement soient utilisés ou non dans les contraintes (2.16). La relaxation linéaire de cette formulation est obtenue en remplaçant les contraintes (2.16) par des contraintes de non négativité.

Notons par π_{z_w} les variables duales associées aux contraintes (2.15). La fonction objectif du sous-problème devient :

$$\min \quad 1 \quad - \sum_{w \in N} \sum_{z_w = 1}^{q_w} \pi_{z_w} a_{z_w e}. \tag{2.17}$$

En effet, le coût de sortie d'un avion est 1 unité et il s'agit donc par la suite de trouver des valeurs de $a_{z_w e}$ qui minimisent la fonction objectif (2.17) et qui correspondent à un chargement réalisable. Ces valeurs de $a_{z_w e}$ sont obtenues en résolvant un problème de plus court chemin dans un réseau avec contraintes de ressources. Ce problème consiste alors à minimiser (2.17) avec les contraintes (2.3)-(2.8).

2.6.2 Le problème du plus court chemin avec contraintes de ressource

Dans cette section, nous décrivons la résolution du sous-problème. Rappelons que le rôle de ce sous-problème est de déterminer le plan de chargement de coût réduit minimal. Si la valeur optimale du sous-problème est non négative, la solution courante du *LP* du *PMR* est optimale pour le *LP* du *PM*. Sinon, cette solution peut être améliorée en ajoutant au moins un plan de chargement de coût réduit négatif au *PMR* qui sera optimisé.

Ce sous-problème est un problème de plus court chemin élémentaire soumis à des contraintes de ressource (*ESPRC* pour *Elementary Shortest Path with Resource Constraints*). C'est un problème $\mathcal{NP}$-difficile au sens fort (Dror, 1994) pour lequel aucun algorithme général efficace n'est connu, voire dans certaines applications le problème demeure sans solution. Des algorithmes pseudo-polynomiaux, spécialisés dans certaines formulations spécifiques, existent cependant lorsque des chemins non-élémentaires sont permis à condition que les fonctions de prolongation soient non décroissantes. Ainsi, des cycles sont autorisés changeant le problème en un plus court chemin avec contraintes de ressource (*SPRC* pour *Shortest Path with Resource Constraints*). Desrochers et Soumis (1988) propose un algorithme pseudo-polynomial et plusieurs variantes pour la résolution de *SPRC*. L'algorithme est aussi décrit dans l'article de Desrosiers *et al.* (1995).

La procédure utilisée pour résoudre le problème *SPRC* se base sur un algorithme d'étiquettage qui maintient un ensemble de chemins défini par un certain nombre d'étiquettes à un nœud du réseau. Une étiquette est un état qui contient une composante coût, mais aussi une composante additionnelle pour chaque ressource. Elle correspond à un chemin partiel allant du nœud *source* jusqu'au nœud où elle est définie. L'algorithme construit alors itérativement dans le réseau des chemins commençant par le nœud *source*, lesquels sont conservés comme des étiquettes une fois

rendus au nœud courant. A la différence des algorithmes classiques d'étiquettage pour les problèmes de plus court chemin sans contraintes, ceux avec contraintes de ressource doivent conserver à chaque nœud des étiquettes multidimensionnelles non dominées. Le critère de dominance est défini dans Desaulniers et *al.* (1998) comme suit : un état s *domine* un autre t si et seulement si chaque composante de s est plus petite ou égale à chaque composante correspondante de l'état t. Ce critère de dominance est exploité par Desaulniers et *al.* pour décroître considérablement le temps de calcul ou la mémoire utilisée dans certaines applications. Ils ont montré, dans le cas particulier où les fonctions de prolongation sont non décroissantes, incluant la fonction de coût, que si l'état s domine l'état t, l'élimination de l'état t est possible à partir du moment où le sous-chemin ne sera présent dans aucune solution optimale du plus court chemin. Il est clair que si les extensions basées sur l'état t sont valides, alors celles basées sur s le sont aussi, et elles sont toujours moins coûteuses ou du moins égales. Donc, l'état t peut être éliminé.

Dans notre modèle, la dernière fonction de prolongation (2.13), celle qui gère la borne inférieure de la contrainte du CG, n'est pas une fonction non décroissante. La proposition suivante permet de détecter les étiquettes dominées.

Proposition 3 *Soit s, t deux états définis à un nœud $i \in N$ avec respectivement comme vecteurs de ressource associés $\mathbf{T}_i^{k,s}$ et $\mathbf{T}_i^{k,t}$ et comme coût respectif c_i^s et c_i^t. L'état s domine l'état t si :*

$$
\begin{aligned}
c_i^s &\leq c_i^t \\
T_i^{k1,s} &\leq T_i^{k1,t} \\
T_i^{k2,s} &= T_i^{k2,t} \\
T_i^{k3,s} &\leq T_i^{k3,t} \\
T_i^{k4,s} &\leq T_i^{k4,t} \\
T_i^{k5,s} &\geq T_i^{k5,t}.
\end{aligned}
$$

Preuve : Supposons que l'extension de l'état t utilisant l'arc $(i,j) \in A^k$ soit un état valide t', une fois rendu au nœud $j \in V^k$. Nous allons montrer que l'état s peut aussi se prolonger, en utilisant l'arc (i,j), à l'état s' avec un coût $c_j^{s'} \leq c_j^{t'}$.

Pour la quatrième ressource, nous avons :

$f^{k4}_{(i,j)}(\mathbf{T}_i^s) = \frac{T_i^{k4,s}T_i^{k2,s}+(T_i^{k1,s}+\alpha_j l_j)m_j}{T_i^{k2,s}+m_j} \leq \frac{T_i^{k4,t}T_i^{k2,t}+(T_i^{k1,t}+\alpha_j l_j)m_j}{T_i^{k2,t}+m_j} = f^{k4}_{(i,j)}(\mathbf{T}_i^t) = T_j^{k4,t'} \leq \overline{g}$ si j correspond à un item.

$f^{k4}_{(i,j)}(\mathbf{T}_i^s) = T_i^{k4,s} \leq T_i^{k4,t} = f^{k4}_{(i,j)}(\mathbf{T}_i^t) = T_j^{k4,t'} \leq \overline{g}$ sinon.

Pour la cinquième ressource associée à la borne inférieure, nous avons :

$\underline{g} \leq T_j^{k5,t'} = f^{k5}_{(i,j)}(\mathbf{T}_i^t) = \frac{T_i^{k2,t}(T_i^{k5,t}-l_j)+m_j(L_B-(1-\alpha_j)l_j)}{T_i^{k2,t}+m_j} \leq \frac{T_i^{k2,s}(T_i^{k5,s}-l_j)+m_j(L_B-(1-\alpha_j)l_j)}{T_i^{k2,s}+m_j} = f^{k5}_{(i,j)}(\mathbf{T}_i^s)$ si j correspond à un item chargé sur le quai.

Pour la cinquième ressource associée à la borne inférieure, nous remarquons que :

$(T_i^{k2}-T_i^{k3})T_i^{k5}+T_i^{k3}(T_i^{k5}-l_j)+m_j(L-(1-\alpha_j)l_j) = T_i^{k2}T_i^{k5}-T_i^{k3}l_j+m_j(L-(1-\alpha_j)l_j).$

En utilisant cette dernière expression, nous avons :

$\underline{g} \leq T_j^{k5,t'} = f^{k5,t}_{(i,j)}(\mathbf{T}_i^t) = \frac{T_i^{k2,t}T_i^{k5,t}-T_i^{k3,t}l_j+m_j(L-(1-\alpha_j)l_j)}{T_i^{k2,t}+m_j} \leq \frac{T_i^{k2,s}T_i^{k5,s}-T_i^{k3,s}l_j+m_j(L-(1-\alpha_j)l_j)}{T_i^{k2,s}+m_j} = f^{k5,s}_{(i,j)}(\mathbf{T}_i^s)$ si j correspond à un item chargé sur la rampe.

S'il est possible de prolonger l'état t vers l'état t', il est aussi possible de prolonger l'état s vers s' avec un coût $c_j^{s'} \leq c_j^{t'}$ et l'étiquette associée à t' peut être éliminée. □

Lors de la résolution du sous-problème, l'algorithme maintient les contraintes opérationnelles de chargement telles la longueur totale des items chargés, le poids total, etc. Par contre, il est possible que des chargements soient générés où le même item apparaisse plus d'une fois. On parle alors de "sur-recouvrement". A remarquer qu'on ne peut pas charger un nombre infini d'items puisque chaque fois qu'un item est chargé, la capacité de l'avion est réduite. Pour améliorer la qualité des bornes inférieures obtenues, l'élimination des cycles, plus précisément les *2-cycles* (cycles de la forme $i \rightarrow j \rightarrow i$ avec $i,j \in N$) peut être incorporée à l'algorithme tout en préservant leur complexité pseudo-polynomiale (Desrochers, Desrosiers et Solomon (1992) ; Kohl et Madsen (1997)). Cependant, l'élimination des cycles d'ordre supérieur est

plus complexe et s'avère moins utile pour des problèmes d'assez grande taille (Kohl, 1995). Ainsi, notre stratégie serait d'avoir une borne inférieure "assez serrée", même si des colonnes comportant des cycles apparaissent dans la solution optimale. C'est le processus de *B&B* qui va prendre la relève pour éliminer les cycles dans le processus d'énumération. En effet, les travaux de Bramel et Simchi-Levi (1993) montrent que la borne inférieure obtenue à partir d'une formulation sous forme de partitionnement du problème est de bonne qualité et que l'écart relatif entre la solution fractionnaire et la solution entière devient arbitrairement petit lorsque le nombre de tâches à couvrir, items à transporter dans notre cas, augmente. Dans ces conditions, le processus de *B & B* est plus efficace.

2.6.3 Stratégies de branchement et plans coupants

La relaxation linéaire de (2.14)-(2.16) est calculée par une procédure de génération de colonnes. A chaque étape de l'algorithme, le rôle du problème maître consiste à chercher la meilleure combinaison de plans respectant les contraintes liantes (2.15). Pour amorcer l'algorithme de résolution, nous considérons seulement les plans triviaux. Cela consiste à charger un item dans chaque chargement. Après avoir résolu ce *LP*, une valeur optimale marginale est associée à chaque item. Ces valeurs vont être utilisées pour obtenir des chargements plus profitables via le sous-problème. Un ensemble de plans de chargement est envoyé au problème maître. Cette procédure est répétée jusqu'à ce qu'aucun plan de chargement ne puisse plus améliorer la solution courante.

La solution de la relaxation linéaire restreinte n'est pas nécessairement entière et l'application du *B&B* standard sur le problème maître avec les colonnes existantes ne nous garantit pas la solution optimale. La méthode de "branch-and-price" consiste à identifier une règle de branchement qui élimine la solution fractionnaire actuelle au

moyen de nouvelles contraintes linéaires. Après chaque branchement effectué sur un nœud de l'arbre de recherche d'un *B&B*, nous devons généralement générer d'autres colonnes ou de montrer l'optimalité de la solution. En outre, la règle doit être compatible avec la génération de colonnes. Plus précisément, suivant les contraintes de branchement imposées, nous devons être en mesure de modifier le sous-problème sans pour autant regénérer des colonnes déjà exclues et rendre les solutions hors d'atteinte. Idéalement, nous devons rencontrer le même niveau de difficulté dans chaque espace de solutions ou dans chaque nœud créé par le branchement dans la recherche d'une bonne solution (Johnson, 1989).

Pour les problèmes dont le problème maître possède une structure de partitionnement, Ryan et Foster (1981) ont développé la règle qui consiste à décider que deux rangées du problème maître doivent être couvertes par la même colonne sur une branche et par deux colonnes différentes sur une autre. Ce mode de branchement tend à équilibrer la recherche dans chaque branche. Barnhart *et al*, (1995) et Desrosiers *et al*, (1984, 1995) se sont inspirés de cette règle. Pour conserver la structure du sous-problème, ils ont appliqué la règle sur les variables de la formulation originale au lieu des variables de la formulation de génération de colonnes. Pour le problème de tournées de véhicules, le branchement illustré suivant a été mis en pratique pour la première fois par Desrochers et Soumis (1989) et la règle est compatible avec les sous-problèmes de type *SPRC* dans un contexte de génération de colonnes. Les décisions de branchement sont prises sur les variables agrégées de flot :

$$X_{ij} = \sum_{k \in K} X_{ij}^k, \quad \forall (i,j) \in \bigcup_{k \in K} A^k, \quad i,j \in N.$$

Ces variables, assimilées à être binaires, déterminent si les deux tâches associées respectivement au nœud i et j sont effectuées l'une après l'autre. Les variables dont la partie fractionnaire sont la plus proche de 0,5 sont préférées pour mieux équilibrer l'arbre de branchement. Ces décisions sont transférées directement dans le

sous-problème. Une décision $X_{ij} = 0$ se traduit par l'omission de l'arc (i, j) du réseau et nous sommes assurés qu'aucune colonne contenant cet arc ne sera plus générée. La décision $X_{ij} = 1$ revient à enlever tous les arcs issus de i ou arrivant à j, à l'exception de l'arc considéré (i, j). De telles décisions ne changent pas la structure du sous-problème, elles réduisent tout simplement leur taille. Pour notre problème, cette règle permet de pallier les problèmes causés par la symétrie entre les avions. En effet, plusieurs plans de chargement peuvent être considérés comme différents de par le rang de l'avion qui utilise le plan, ce qui fait qu'en échangeant les rôles de deux avions, nous avons deux mêmes plans associés à des variables de valeurs différentes. Lorsque nous excluons un plan dont la variable associée est fractionnaire à un nœud de l'arbre de branchement, ce plan peut réapparaître à un autre nœud sous une autre variable et sous une autre valeur. Remarquons qu'avec cette règle, un seul avion parmi tous ceux qui utilisent le même plan est retenu parce que la somme de ces variables sur les avions considérés prend une valeur binaire, par conséquent, soit nous retrouvons deux items, associés respectivement au nœud i et j, chargés l'un après l'autre sur un même avion, soit nous ne retrouvons pas les deux items simultanément nulle part.

Dans la modélisation de *ALP*, rappelons que des nœuds intermédiaires ont été ajoutés dans le réseau, entre autres pour réduire la taille du réseau. De ce fait, nous ne pouvons pas utiliser telle quelle la méthode ci-dessus car le fait d'enlever un ou plusieurs arcs ne garantit plus l'optimalité de la méthode. Les décisions de branchement vont être prises sur les valeurs des inter-tâches (les items sont considérés comme des tâches). Ces valeurs représentent la quantité chargée entre deux items consécutifs. Comme d'après les contraintes (2.15), tous les items doivent être chargés exactement une seule fois, toutes les valeurs des inter-tâches doivent être binaires. Ainsi, lorsque la solution contient une inter-tâche de valeur fractionnaire, deux branches sont créées : une où la valeur du flot est mise à 1 et une autre où cette valeur est égale à 0. L'algorithme du plus court chemin doit être modifié pour assurer que la

procédure de dominance conserve l'optimalité de la solution. Pour chaque nœud de l'arbre de branchement, la décision est imposée dans le sous-problème en conservant plusieurs listes d'étiquettes à chaque nœud du réseau. Chaque liste regroupe les étiquettes correspondantes aux plans partiels dont le dernier item chargé est le même. La dominance entre les étiquettes s'effectue maintenant à l'intérieur de chaque liste. Donc, tous les sous-problèmes conservent la même structure de plus court chemin et le même nombre de contraintes de ressource. Cette version plus générale de la règle de branchement de Desrochers et Soumis a été implantée par Dumas et *al.* (1991) dans le contexte de problèmes de ramassage et de livraison, et par Haase et *al.* (2001) pour un problème de fabrication simultanée de tournées d'autobus et d'horaires de chauffeurs.

Une coupe est aussi utilisée pour renforcer la relaxation linéaire à travers l'arbre de recherche. La valeur Z_{LP} du LP est arrondie à l'entier supérieur pour former la coupe suivante :

$$\sum_{k \in K} \sum_{w \in N^k} X^k_{O^k B_{w1}} = \sum_{e \in \Omega} \theta_e \geq \lceil Z_{LP} \rceil .$$

Cette contrainte a la même forme qu'une contrainte de branchement et elle est tranférée dans la fonction objectif du sous-problème en soustrayant le coût réduit par une constante égale à la variable duale correspondant à la contrainte. Cette coupe relative au nombre d'avions-cargos améliore la borne inférieure du problème maître et elle a été utilisée avec succès, entre autres, par Desrosiers et *al.* (1984) et récemment par Kohl (1995) pour des problèmes de routage de véhicules.

Pour résoudre de façon exacte les instances présentées dans la section suivante, l'arbre de recherche du *B&B* est exploré en utilisant la procédure de "profondeur d'abord" (*depth-first search*) c'est-à-dire en sélectionnant le nœud le plus récemment créé. L'avantage de cette exploration est double : obtenir rapidement une solution réalisable et limiter au minimum la mémoire utilisée. L'inconvénient de ce type de choix est de

ne pas tenir compte de la fonction d'évaluation c'est-à-dire qu'on peut être amené à traiter des sous-ensembles dont l'évaluation est médiocre et qui ont peu de chance de contenir une solution optimale. Nous avons exploré en premier la branche où le flot est mis à 1. Le successeur de ce nœud est traité par la suite, jusqu'à ce que la procédure de recherche rencontre un autre nœud où la décision à prendre implique un avion en plus. La direction de l'exploration revient sur ses pas (traduction libre de "backtracking") en remontant d'un niveau, cherche la branche où le flot est mis à 0 et continue sa descente dans cette direction. Dans chaque branche, la décision est imposée dans le sous-problème. Une fois qu'une solution est trouvée, la procédure de recherche "meilleure borne d'abord" (*best-first search*), c'est-à-dire en sélectionnant le nœud le plus promettant et dont l'évaluation est la plus petite, est adoptée pour compléter l'exploration. L'algorithme s'arrête lorsque la solution optimale est trouvée et prouvée.

2.7 Résultats numériques

L'algorithme décrit dans la section précédente est testé sur un problème réel et des problèmes générés aléatoirement. Le problème réel est tiré de l'article de Ng (1992) et est rapporté dans le tableau 2.6. Les caractéristiques de chaque item sont celles données par le Département de la Défense Nationale du Canada (DND). Pour évaluer l'impact des priorités d'envoi sur la performance de l'algorithme, nous avons ajouté arbitrairement des nombres représentant des priorités entre 1 et 4 dans certains scénarios, pour chaque item. Les 322 items doivent être transportés par des *Hercules CC130* pour lesquels la longueur du quai est de 492 pouces et la longueur de la rampe est de 132. Le poids maximal permis est de 195 900 livres sur le quai, et de 5 000 sur la rampe. Les items 2, 7, 11, et 16 sont les seuls pouvant être chargés sur la rampe.

Tous les tests ont été effectués sur un ordinateur *SunEnterprise*10000 (400 Mhz,

Tableau 2.6 – Les données réelles de Ng (1992)

i	Qté	l_i	m_i	p_i	Description de l'item
1	40	158	3500	1	$\frac{1}{4}$ Ton Utility Truck
2	48	124	4000	1	1 $\frac{1}{4}$ Ton Cargo Truck
3	3	279	19500	2	2 $\frac{1}{2}$ Ton Bowser
4	47	267	17000	3	Medium Logistics Vehicle Wheeled 2 $\frac{1}{2}$ ton
5	2	204	21300	3	5 Ton Truck
6	11	232	24800	4	Armoured Personel Carrier M113
7	34	109	1100	3	$\frac{1}{4}$ Ton Trailer
8	18	147	2900	2	$\frac{3}{4}$ Ton Trailer
9	19	166	5400	1	1 $\frac{1}{2}$ Ton Trailer
10	2	158	12000	2	Rough Terrain Fork Lift
11	4	109	1000	3	Herman Nelson Heater
12	2	271	22500	1	Front Head Loader
13	22	104	20500	4	Armored Vehicle General Purpose
14	6	166	5100	2	Howitzer 105
15	9	58	3000	2	Pallet 54" × 88"
16	11	109	1000	3	Triwalls
17	6	252	8000	1	Twin Huey CH135
18	3	388	2000	3	Kiowa CH136
19	34	224	9400	1	1 $\frac{1}{4}$ Ton Command Post
20	1	310	33800	4	5 Ton Wrecker

64Go, unix, gcc compiler). Pour résoudre les problèmes, nous avons utilisé l'optimiseur GENCOL développé à Montréal. Nous avons choisi comme variable à fixer celle dont le flot est le plus grand entre 0,75 et 1 et, en cas de litige, celle dont un des nœuds est associé à un long item. Ce critère de choix répond au besoin de descendre au fond de l'arbre de recherche pour chercher les solutions prometteuses le plus tôt possible. Avec la stratégie de recherche adoptée, nous nous attendons à ce que la première solution trouvée soit la bonne. De plus, plusieurs colonnes dont le coût réduit est négatif sont remontées à chaque itération dans le problème maître pour accélerer l'algorithme. Cette stratégie a été suggérée par Vanderbeck (1994) lorsque la résolution du sous-problème prend suffisamment de temps tel que c'est le cas dans notre application. Elle vise surtout à obtenir un équilibre entre l'effort de calcul effectué par le problème maître et le sous-problème.

2.7.1 Nombre de séries

Le nombre de séries que nous avons considéré dans chaque réseau a clairement un impact sur la difficulté du problème. Nous pouvons facilement calculer *a priori* le nombre de séries suffisant pour résoudre à l'optimalité le problème tel que décrit dans le modèle ; cela correspond au nombre d'items d'un chargement en alternant le moins long et le moins long suivant selon les disponibilités de chacun. Notons que pour l'instance donnée, la longueur d'un item est plus contraignante que son poids. Pour les données du tableau 2.6, nous avons besoin d'au plus six séries sur le quai et d'une seule sur la rampe (obtenues à partir des items 15 et 13). Le tableau 2.7 présente les résultats à partir des données de Ng (1992) lorsque le nombre de séries varie de deux à six pour le quai, et celui de la rampe est fixé à un. La solution optimale est garantie uniquement lorsque le nombre de séries est égal à six, même si cette solution est obtenue à partir de trois séries.

Pour ces tests, l'intervalle admissible ou la "fenêtre" du CG est fixé à (551,564) lequel correspond à la contrainte réelle pour un *Hercules C130* (Richardson , 1993). Nous étudierons un peu plus loin l'effet de cette fenêtre sur la qualité de la solution. Aucune priorité d'envoi n'est imposée, c'est-à-dire que, chaque chargement peut contenir n'importe quel item du groupe des 322 à transporter. Pour chaque problème, nous mentionnons le temps de résolution (CPU en secondes), le temps de résolution de la relaxation linéaire (TiRel en secondes), la valeur optimale (Sol), la valeur de la relaxation linéaire (Relax), le saut d'intégralité (Gap), le nombre de nœuds explorés (B&B), le nombre de colonnes générées (ColGen), le nombre d'arcs et de nœuds pour décrire le problème (Arcs et Nœuds) et la mémoire maximale requise pour le résoudre (Mem en Mb).

Ces premiers résultats montrent que nous pouvons résoudre à l'optimalité le problème du DND en moins de 6 minutes, lesquelles sont clairement acceptables pour une

Tableau 2.7 – Différents nombres de séries sur le quai et une seule fixée sur la rampe

model	2 Series	3 Series	4 Series	5 Series	6 Series	SSNœud
CPU	89	204	215	271	319	4542
TiRel	76	143	151	188	190	1879
Sol	94	92	92	92	92	92
Relax	93.1	92.0	92.0	92.0	92.0	92.0
Gap %	0.9	0.0	0.0	0.0	0.0	0.0
B&B	190	208	202	181	210	410
ColGen	6065	10100	9001	9128	9383	5717
Arcs	2695	4061	5427	6793	8159	632766
Nœuds	843	1205	1567	1929	2291	2040
Mem	8.4	28.7	51.4	57.1	57.8	161.0

phase de planification. Le temps CPU, la taille du réseau et la mémoire requise pour résoudre le problème sont les facteurs qui croissent avec le nombre de séries considérées. Les résultats montrent que trois séries sont suffisantes pour trouver le nombre optimal de "chalks" (92) pour cet ensemble d'items. Cette solution requiert 18 voyages de moins que celle de Ng. Avec les coûts estimés par Ng, (à peu près $110 000/voyage) le coût épargné grâce à cette solution est de $2 millions par rapport à celle de Ng et de $3 millions comparée à celle de la première solution proposée par les *loadmasters*. Ng résout un problème de recouvrement en sélectionnant 38 plans de chargement. L'approche de génération de colonnes permet de considérer implicitement tous les plans réalisables (9 383 plans ont été générés) ce qui explique la qualité de la solution trouvée.

La dernière colonne montre le résultat du même problème en utilisant le réseau "naïf" de la section 2.2.1 tel qu'il est décrit selon les figures 2.1 (a) et 2.2 (a) . Ainsi, six séries sont mises sur le quai et une seule sur la rampe, en outre aucun nœud intermédiaire ne se trouve entre deux items quelconques. Nous remarquons dans ce cas que le nombre d'arcs du réseau est très grand. Ce résultat montre l'efficacité d'avoir un réseau approprié, et lequel n'explose pas à mesure que la taille du problème augmente.

2.7.2 Variation du nombre d'items et de la contrainte du CG

Les tests suivants (tableaux 2.8, 2.9 et 2.10) concernent l'impact de la quantité d'items transportés, la rigueur de la contrainte du CG et l'utilisation des priorités d'envoi. Ces priorités ont été attribuées à la main. A partir des données de Ng et avec des scénarios comportant des priorités (d'abord aucun et ensuite, avec 4 priorités d'envoi comme indiqués au tableau 2.8), nous avons résolu 4 problèmes correspondant au nombre d'items considérés. Le premier problème (première colonne marquée "Une") est celui de Ng, le second (colonne "Deux") est le double de la taille du problème de Ng (le nombre d'items pour chaque type est multiplié par deux) et ainsi de suite pour la colonne "Trois" et "Quatre". L'intervalle du CG reste fixé à (245,737) lequel revient à relaxer la contrainte du CG (le CG peut être n'importe où sur le quai), le second (551,564) est l'actuel CG accepté pour les *Hercules* tandis que le troisième (559,563) est plus serré.

Tableau 2.8 – Intervalle large pour le CG : (245,737)

Priorité	1 Priorité				4 Priorités			
modèle	Une	Deux	Trois	Quatre	Une	Deux	Trois	Quatre
CPU	202	1963	8715	22590	77	655	2963	8646
TiRel	71	544	1979	5247	19	121	472	1241
Sol	92	184	275	367	101	201	302	402
Relax	91.6	183.1	274.6	366.2	101.0	201.0	302.0	402.00
Gap %	0.5	0.5	0.1	0.2	0.0	0.0	0.0	0.0
B&B	289	564	864	1149	242	476	743	959
ColGen	4293	9356	14150	19316	3026	6422	9515	13039
Arcs	8159	14273	20387	26501	10476	19266	28056	36846
Nœuds	2291	4329	6367	8405	2291	4329	6367	8405
Mem	76.4	155.0	241.0	334.3	11.0	26.1	46.2	70.7

Nous observons que le temps CPU croît exponentiellement avec le nombre d'items transportés même si les autres paramètres (nombre d'itérations, colonnes générées,

Tableau 2.9 – Intervalle moyen pour le CG : (551,564)

Priorité	1 Priorité				4 Priorités			
modèle	Une	Deux	Trois	Quatre	Une	Deux	Trois	Quatre
CPU	319	2410	9589	23680	83	805	3588	8335
TiRel	191	1218	4366	9719	22	134	451	1078
Sol	92	184	276	368	106	212	318	424
Relax	92.0	184.0	276.0	368.0	106.0	212.0	318.0	424.0
Gap	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B&B	210	443	631	914	198	468	702	932
ColGen	9383	26127	56548	59151	3077	6771	9828	13899
Arcs	8159	14273	20387	26501	10476	19266	28056	36846
Nœudss	2291	4329	6367	8405	2291	4329	6367	8405
Mem	57.8	119.7	189.2	263.2	10.7	25.9	45.9	76.2

nombre de nœuds, etc.) indiquent que la difficulté du problème semble augmenter linéairement. Cela est dû aux algorithmes qui résolvent le problème maître et le sous-problème. De plus, le nombre d'étiquettes qui gèrent les chemins partiels augmente rapidement en fonction du nombre d'items transportés. Rappelons que nous pouvons adapter notre algorithme pour résoudre les problèmes dont le CG de chaque item n'est pas localisé en son centre ($\alpha = \frac{1}{2}$). Dans ce cas, chaque item est dédoublé dans le réseau pour modéliser les deux positions possibles (par devant ou par derrière) de chargement. Ainsi, la colonne marquée "Deux" nous donne un ordre de grandeur du temps CPU et de la mémoire requis si nous modélisons de telles situations avec les mêmes données.

Les problèmes contenant des priorités d'envoi sur les items sont toujours résolus plus rapidement, bien que la taille totale des trois sous-réseaux soit plus grande que celle avec une seule priorité (même nombre de nœuds mais avec moins d'arcs). Notre modélisation sur les priorités d'envoi implique seulement que la solution optimale requiert un nombre plus grand ou égal d'avions que celle sans priorité. L'impact des priorités d'envoi sur le nombre minimal de chargements trouvé est clairement illustré par les tableaux 2.8, 2.9 et 2.10 ; par exemple avec un intervalle moyen sur le CG, nous

Tableau 2.10 – Intervalle serré pour le CG : (559,563)

Priorité	1 Priorité				4 Priorités			
modèle	Une	Deux	Trois	Quatre	Une	Deux	Trois	Quatre
CPU	216	1503	5860	12770	93	869	3776	9747
TiRel	164	876	2893	6235	27	137	406	1072
Sol	97	194	291	388	108	216	324	432
Relax	96.9	193.7	290.6	387.4	108.0	216.0	324.0	432.0
Gap	0.1	0.1	0.1	0.1	0.0	0.0	0.0	0.0
B&B	199	491	695	901	185	395	603	787
ColGen	7567	17863	33949	46319	3276	6735	10002	13665
Arcs	8159	14273	20387	26501	10476	19266	28056	36846
Nœuds	2291	4329	6367	8405	2291	4329	6367	8405
Mem	46.7	98.0	157.7	223.0	10.7	25.2	44.0	68.4

avons besoin de 14 avions additionnels par rapport au cas avec un intervalle large pour tranporter les 322 items. L'impact sur le CG n'est pas non plus négligeable. La réduction de la contrainte du CG de (551,564) à (559,563) requiert cinq avions additionnels si aucune priorité d'envoi n'est considérée et deux sinon.

2.7.3 Problèmes aléatoires

Nous concluons cette section concernant les résultats numériques en résolvant 30 instances générées aléatoirement. L'intervalle du CG est fixé à (551,564) et aucune priorité n'est imposée. Les instances sont générées de telle sorte qu'elles représentent des problèmes réels. Le nombre de types différents est fixé à 20 pour les 30 problèmes. La quantité pour chaque type d'items est un nombre aléatoire entre 1 et 50. La longueur de chaque type est entre 50 et 300 avec une forte probabilité dans l'intervalle [100 ;150] et le poids est entre 10 et 350 avec une forte probabilité dans l'intervalle [10 ;100]. Les détails sur la génération de ces données sont décrits à l'annexe.

Les résultats sont détaillés dans les tableaux 2.11 et 2.12. La moyenne et l'écart-type sont donnés dans les deux premières colonnes (MOY et E.T) du tableau 2.11.

Tableau 2.11 – Problèmes aléatoires (partie 1)

	MOY	E.T	N1	N2	N3	N4	N5	N6
CPU	1628	2015	340	224	763	4748	2197	552
TiRel	1120	1716	111	152	359	4266	1099	300
Sol	145	26	135	120	151	103	151	140
Relax	144.2	25.7	135.0	119.4	151.0	102.6	150.8	139.3
Gap	0.3	0.2	0.0	0.5	0.0	0.4	0.1	0.5
B&B	377	74	297	266	366	398	441	355
ColGen	9959	7873	4261	4237	6681	13511	9025	5445
Arcs	11059	2281	8249	7460	9331	10922	13048	9303
Nodes	3272	698	2411	2143	2770	3212	3919	2764
Mem	270.8	346.4	59.2	19.0	191.9	913.8	291.7	55.2
Items	507	70	426	388	504	469	587	487
Itm/chalks	3.5	2.7	3.2	3.2	3.3	4.6	3.9	3.5
Series			5+1	5+0	5+1	6+1	6+1	5+1
NbMaxIt			5	4	5	6	7	4

	N7	N8	N9	N10	N11	N12	N13	N14
CPU	510	1282	1145	704	1614	1949	227	1066
TiRel	192	950	385	318	1058	1685	78	516
Sol	157	146	200	133	195	120	131	146
Relax	156.9	145.9	199.9	132.5	194.3	119.4	131.0	145.6
Gap	0.1	0.1	0.1	0.4	0.4	0.5	0.0	0.2
B&B	324	299	462	392	422	323	282	420
ColGen	5475	5549	7092	6277	10722	9790	3976	6972
Arcs	10686	14035	11407	10858	15527	10183	8089	12073
Nœuds	3130	4058	3462	3184	4647	2959	2361	3594
Mem	165.7	358.1	95.6	223.1	243.7	417.1	48.9	89.2
Items	470	463	640	483	617	445	410	534
Itm/chalks	3.0	3.2	3.2	3.6	3.2	3.7	3.1	3.7
Series	6+1	8+1	5+1	6+1	7+1	6+1	5+1	6+1
NbMaxIt	5	5	6	6	5	5	5	5

	N15	N16	N17	N18	N19	N20	N21	N22
CPU	1144	823	1394	4370	1212	458	350	892
TiRel	780	522	815	3540	724	141	155	615
Sol	137	112	152	124	111	162	159	118
Relax	136.5	112.0	151.2	123.5	110.8	161.8	158.4	117.5
Gap	0.3	0.0	0.5	0.4	0.2	0.2	0.4	0.5
B&B	367	315	460	470	407	379	349	353
ColGen	9523	19645	9934	30464	11951	4867	4329	10027
Arcs	13094	8379	12667	10692	10726	9461	10848	8940
Nœuds	3836	2451	3792	3227	3145	2815	3184	2643
Mem	191.9	113.4	167.0	752.0	273.6	32.2	57.5	196.7
Items	504	440	564	571	462	505	483	467
Itm/chalks	3.7	3.9	3.7	4.6	4.2	3.1	3.0	4.0
Series	7+1	5+1	6+1	5+1	6+1	5+1	6+1	5+1
NbMaxIt	5	5	4	6	5	4	4	5

Tableau 2.12 – Problèmes aléatoires (partie 2)

	N23	N24	N25	N26	N27	N28	N29	N30
CPU	175	9782	1021	338	472	1581	5781	1736
TiRel	43	7859	596	171	276	437	4802	664
Sol	188	114	154	128	174	166	123	186
Relax	188.0	113.9	154.0	127.4	173.2	165.5	122.9	185.3
Gap	0.0	0.1	0.0	0.5	0.5	0.3	0.1	0.4
B&B	250	429	313	267	384	492	498	526
ColGen	3142	27489	8068	4177	5209	7089	35321	8516
Arcs	8821	14471	13017	8966	9917	15556	13027	14048
Nœuds	2600	4300	3812	2555	2967	4660	3912	4254
Mem	22.1	1708.1	145.5	81.1	39.0	332.8	740.3	97.4
Items	467	557	499	382	545	607	585	641
Itm/chalks	2.5	4.9	3.2	3.0	3.1	3.7	4.8	3.5
Series	5+1	7+1	7+1	6+1	5+1	7+1	6+1	4+1
NbMaxIt	4	6	5	4	4	6	6	5

Quatre lignes additionnelles sont données dans ces tableaux : le nombre d'items transportés (Items), le nombre moyen d'items par avion (Itm/chalk), le nombre de séries utilisées dans le modèle (Séries) et le nombre maximal d'items chargés par un avion dans la solution optimale (NbMaxIt). Nous notons que le nombre d'items transportés est toujours plus grand que les 322 items considérés par Ng et que le nombre moyen d'items par avion est à peu près le même. Ce sont certainement des facteurs qui indiquent la difficulté du problème et qui sont confirmés par les quatre problèmes les plus difficiles (N4, N18, N24 et N29). Les résultats indiquent que notre approche est davantage efficace quand le nombre d'items par avion reste relativement petit (c'est-à-dire moins de six, comme c'est le cas pour le problème du DND). Quand ce nombre augmente, la mémoire requise pour résoudre le problème devient trop importante, en raison principalement du nombre exponentiel d'étiquettes non dominées dans l'algorithme pour résoudre le plus court chemin avec contraintes de ressource. Enfin, nous avons observé la propriété IRUP (*Integer Rounding Up Property*) (voir section 2.2.2, proposition 1) après le prétraitement proposé et ce, pour *tous* les problèmes résolus incluant celui de Ng et ses dérivés, et les problèmes

générés aléatoirement.

2.8 Conclusion

Dans ce chapitre, nous avons présenté un modèle de résolution du problème de chargement des avions-cargos entre un point de collecte et un point de livraison. Les considérations des contraintes opérationnelles du problème, et pour la première fois le centre de gravité de chaque item et de l'ensemble avion/cargos rendent le modèle proposé beaucoup plus réaliste que ceux qu'on retrouve dans la littérature. Le modèle peut supporter certaines extensions qui ont été présentées comme des cas spéciaux, entre autres le transport des passagers, les considérations des priorités d'envoi des items, le cas des items dont la densité de masse est non uniforme et les contraintes de préséance. L'approche optimale proposée se base sur la méthode de génération de colonnes. Des procédures de prétraitement des données ont été effectuées pour améliorer la borne inférieure. Des coupes ont aussi été proposées et les stratégies de branchement pour trouver la borne supérieure sont celles de Desrochers et Soumis (1989). Les résultats numériques montrent que le problème réel de Ng (1992) et ses dérivés, ainsi que les problèmes aléatoires, ont été résolus à l'optimalité. Des études sur le nombre d'items transportés et la variation de la fenêtre de faisabilité du CG ont aussi été réalisés. Cependant, nous avons noté quelques considérations qui feront l'objet du chapitre suivant, telles l'amélioration du temps CPU pour résoudre chaque problème et/ou la réduction de la mémoire requise pour sa résolution.

CHAPITRE 3 : L'AGRÉGATION DES CONTRAINTES DE DEMANDE

3.1 Introduction

Dans ce chapitre, nous nous intéressons toujours au problème de chargement des avions-cargos où tous les items se retrouvent en un seul point de collecte et pour une seule destination de livraison. Nous étudions l'agrégation des contraintes liant certains items. Plus précisément, nous allons grouper les items possédant les mêmes caractéristiques (longueur, poids, priorité). La demande est maintenant un nombre plus grand ou égal à un. Ainsi, lorsque 40 Jeeps de même priorité doivent être chargées, ces 40 exemplaires de Jeeps sont spécifiques dans l'approche du chapitre précédent et 40 contraintes de couverture sont associées à cette demande dans la formulation. Dans un premier modèle, une seule contrainte de couverture est associée à ces Jeeps avec une demande de 40 unités. De ce fait, les Jeeps figurant dans un plan de chargement quelconque sont des génériques ou des représentants des Jeeps de l'ensemble plutôt que des Jeeps spécifiques. Dans un deuxième modèle, l'unique contrainte de couverture, associée à ces Jeeps dans le premier modèle, est partitionnée en un nombre minimal de contraintes ayant une unité comme demande et, éventuellement, une autre contrainte avec une demande plus grande que l'unité.

Chaque modèle reste toujours celui d'un réseau multicommodités, mais les nœuds et les arcs de chaque réseau respectif vont subir des modifications. Nous présentons, pour les deux modèles, une formulation utilisant des variables réelles et en nombres entiers, pas seulement binaires. La décomposition de Dantzig-Wolfe est appliquée à ces nouvelles formulations pour fournir la borne inférieure de la relaxation linéaire. La

structure du sous-problème reste inchangée : celle d'un problème de plus court chemin avec contraintes de ressource. Une borne inférieure de la relaxation linéaire est trouvée rapidement, mais la principale difficulté se situe dans la recherche de la solution entière. Un certain nombre d'applications basées sur l'agrégation des contraintes sont exposées dans la revue de la littérature. Des heuristiques et des méthodes exactes sont proposées pour résoudre ces problèmes. Cependant, l'approche de résolution basée sur le problème de plus court chemin avec contraintes de ressource ne nous permet pas de bénéficier directement des algorithmes exacts trouvés dans la littérature pour traiter l'agrégation des contraintes. Le premier modèle est résolu par le critère de Vanderbeck (2000), spécialisé pour les problèmes de découpe. L'heuristique a été implantée et testée sur les mêmes données que celles du précédent chapitre. Les résultats obtenus pour l'ensemble des tests montrent que tous les problèmes ont été résolus à l'optimalité.

Dans l'optique d'une approche optimale, le deuxième modèle est alors proposé et une nouvelle règle de branchement optimale compatible avec la génération de colonnes est développée. La règle s'inspire de celle de Desrochers et Soumis (1989) et elle élimine des problèmes de symétrie, améliorant ainsi l'efficacité des décisions de branchement. Nous avons aussi implanté, testé et comparé notre méthode avec celle de l'approche non agrégée du premier chapitre sur les mêmes données. Les résultats obtenus donnent des solutions optimales pour tous les problèmes.

L'organisation de ce chapitre est la suivante. Dans la section suivante, nous présentons le modèle agrégé. Nous commençons par décrire le nouveau réseau en mettant l'accent sur les différents changements opérés par rapport à l'original. La formulation mathématique du problème est ensuite posée. Dans la section 3.3, quelques méthodes de résolution du problème maître sont présentées. Nous discutons en particulier de deux méthodes de résolution : celle résolvant les problèmes de découpe et celle résolvant les problèmes de tournées de véhicules avec préemption de la demande. Dans

la section 3.4, nous présentons le critère de Vanderbeck et les règles de branchement correspondantes pour résoudre le modèle. Les résultats de tests numériques sont présentés à la section 3.5. La section 3.6 est consacrée à la discussion d'une nouvelle approche. Le réseau correspondant est détaillé ainsi que la formulation et les règles de branchement pour trouver la solution entière. Les résultats des tests effectués avec la méthode proposée sont présentés à la section 3.7. Par la suite, des extensions des méthodes de branchement implantées sont proposées à section 3.8. Enfin, la dernière section contient notre conclusion.

3.2 Le modèle agrégé

L'agrégation des contraintes liant les items de même type, c'est-à-dire en considérant chaque groupe d'items de longueurs, de poids et de priorités égaux comme un seul type d'item dont le nombre correspond à la demande, implique des modifications sur le réseau initial. Ces modifications se situent uniquement à l'intérieur de chaque groupe d'items. Le réseau que nous présentons ci-après est obtenu à partir du précédent, en gardant certains nœuds et en modifiant les connexions entre ces nœuds dans un groupe quelconque. Les rôles de ces nœuds et arcs vont être explicités ultérieurement. Avec ces transformations ou agrégations des contraintes, nous avons un modèle agrégé. La formulation correspondante du problème va être précisée dans la suite ainsi que le problème maître et le sous-problème.

3.2.1 Les changements opérés sur les nœuds et les arcs

Rappelons que, pour décrire les plans de chargement intéressants, nous avons besoin d'un réseau que nous associons à un avion-cargo $k \in K$. Chaque item à transporter

est en correspondance avec un nœud dans le réseau du modèle initial et ce nœud est reproduit autant de fois que le nombre de séries $s \in S$ le permet sur le quai et sur la rampe pour décrire les différentes possibilités de placement de l'item dans l'avion. Au lieu d'énumérer tous les exemplaires $z_w, z_w = 1, ..., q_w$ d'un groupe $w, w \in N$, nous avons besoin en tout de $r_w = \min\{\lfloor \frac{L_B}{l_w} \rfloor, \lfloor \frac{M_B}{m_w} \rfloor, q_w\}$ représentants sur le quai et de $p_w = \min\{\lfloor \frac{L_R}{l_w} \rfloor, \lfloor \frac{M_R}{m_w} \rfloor, q_w\}$ sur la rampe. Ces nombres désignent le nombre maximal d'items pour ce groupe pouvant être mis respectivement sur le quai et sur la rampe. A ces représentants de l'item sont respectivement associés r_w et p_w nœuds et ces derniers sont reproduits autant de fois que le nombre de séries $s \in S$ le permet sur le quai et sur la rampe. Les nœuds $B^k_{ws}, w = 1, .., |N^k|$ sont éliminés mais tous les autres nœuds intermédiaires du réseau précédent sont gardés.

Le nombre d'arcs du réseau est réduit. Les arcs du type 3,4,5 et 6 du chapitre 2, section 2.3 sont tous conservés. Par contre, ceux du type 1 sont modifiés et ceux du type 2 sont enlevés. Par symétrie et sans perte de généralité, nous supposons que le premier représentant est toujours le premier à être chargé, pour les items appartenant au même groupe $w \in N$. Ainsi, pour les arcs du type 1, les arcs $(O^k, I^k_{w\,1\,1}), w = 1, .., |N^k|$ (voir Figure 3.1) sont définis pour permettre le chargement à la première position du premier représentant du groupe w sur le quai. Les arcs $(BR^k, I^k_{w\,1\,s})$ sont définis pour la sélection du premier représentant de l'item w chargé sur la rampe. Ces arcs relient le *quai-rampe* avec la première série sur la rampe.

Un coût de 1 est associé à tous les arcs $(O^k, I^k_{w\,1\,1}), w = 1, ..., |N^k|$, et un coût de 0 pour tous les autres arcs. Le coût d'une unité est chargé pour chaque unité de flot qui passe par l'arc. Cela correspond à la création d'un nouveau plan de chargement.

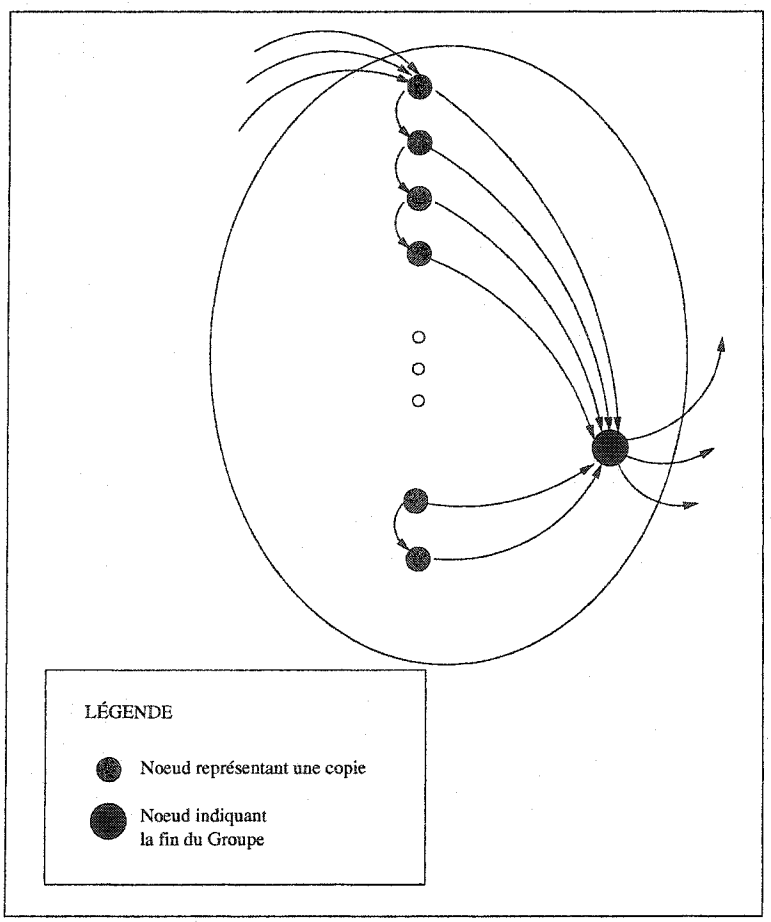

Figure 3.1 – Un groupe d'items de même type

3.2.2 La formulation mathématique du modèle agrégé

En utilisant les mêmes notations que dans la formulation originale, la formulation du modèle agrégé de *ALP* s'écrit, dans ce cas, de la façon suivante :

$$z_{IP} = \min \sum_{k \in K} \sum_{w \in N^k} X^k_{O^k I_{w\,1\,1}} \tag{3.1}$$

sujet aux contraintes :

$$\sum_{k \in K} \sum_{s \in S} \sum_{z_w=1}^{r_w} \sum_{(i,j) \in A^k : j = I^k_{w\,z_w\,s}} a^k_{w,ij} X^k_{ij} = q_w, \forall w \in N \tag{3.2}$$

$$\sum_{j:(O^k,j) \in A^k} X^k_{O^k j} = \sum_{j:(j,D^k) \in A^k} X^k_{jD^k} = 1, \forall k \in K \tag{3.3}$$

$$\sum_{j:(i,j) \in A^k} X^k_{ij} - \sum_{j:(j,i) \in A^k} X^k_{ji} = 0, \forall k \in K, \forall i \in V^k \tag{3.4}$$

$$X^k_{ij} \in \{0,1\}, \forall k \in K, \forall (i,j) \in A^k \tag{3.5}$$

$$X^k_{ij}(f^{kr}_{ij}(\mathrm{T}^k_i) - T^{kr}_j) = 0, \forall k \in K, \forall r \in R, \forall (i,j) \in A^k \tag{3.6}$$

$$a^{kr}_i \leq T^{kr}_i \leq b^{kr}_i, \forall k \in K, \forall r \in R, \forall i \in \{O^k, D^k\} \tag{3.7}$$

$$a^{kr}_i \Big(\sum_{j:(i,j) \in A^k} X^k_{ij} \Big) \leq T^{kr}_i \leq b^{kr}_i \Big(\sum_{j:(i,j) \in A^k} X^k_{ij} \Big), \forall k \in K, \forall r \in R, \forall i \in V^k. \tag{3.8}$$

La fonction objectif (3.1) représente le nombre d'avions-cargos requis pour transporter tous les items. Les contraintes (3.2) impliquent que les q_w exemplaires de l'item de type $w \in N$ soient transportés. Ces dernières s'interprètent comme la contribution à la même tâche de toutes les copies de tous les représentants. Nous avons créé plusieurs copies des représentants placées dans plusieurs positions du réseau, dans le but de modéliser toutes les possibilités de placement dans l'avion. Dans la sommation sur le nombre d'exemplaire z_w, la borne supérieure est r_w car $p_w \leq r_w$. Les contraintes (3.3)-(3.8) sont les mêmes que les contraintes (2.3)-(2.8).

Le problème maître et le sous-problème, issus de la décomposition de Dantzig-Wolfe de cette formulation, sont maintenant décrits.

3.2.3 Le problème maître et le sous-problème

Le coût de sortie d'un avion $k \in K$ est de 1 unité. Soit θ_e la variable indiquant le nombre de fois que le plan de chargement $e \in \Omega$ a été utilisé. Notons par a_{we}^k les coefficients entiers qui représentent le nombre de fois qu'un item de type w est chargé ou non dans l'avion k pour le plan de chargement e. Puisque les avions sont identiques, posons $a_{we}^k = a_{we}$. A partir de ces coefficients, le problème maître s'écrit :

$$\min \sum_{e \in \Omega} \theta_e \tag{3.9}$$

sujet aux contraintes :

$$\sum_{e \in \Omega} a_{we}\theta_e = q_w, \forall w \in N \tag{3.10}$$

$$\theta_e \geq 0, \forall e \in \Omega \tag{3.11}$$

$$\theta_e \in \mathbb{Z}^+, \forall e \in \Omega. \tag{3.12}$$

Le problème maître est écrit en termes de plans de chargement et la fonction objectif minimise le nombre de chargements ou, en d'autres termes, le nombre d'avions requis. Les contraintes (3.10) impliquent que pour chaque type d'item, tous les exemplaires soient chargés suivant des plans appropriés e. Et enfin, nous exigeons que les plans de chargements soient utilisés un nombre entier de fois dans les contraintes (3.12). La relaxation linéaire de cette formulation est obtenue en ne considérant pas les contraintes (3.12).

Notons que dans le réseau utilisant la première formulation, nous associons une demande d'une unité à chaque contrainte de couverture d'un item spécifique à charger, avec lequel une variable duale est en correspondance. Dans la formulation agrégée, une demande de q_w est associée à l'item $w \in N$ et un certain nombre de nœuds situés sur le quai et/ou sur la rampe représentent l'item. Une seule variable duale est en

correspondance avec la contrainte et il s'ensuit que les arcs qui partent des nœuds associés, indépendamment de leur position $s \in S$ dans le réseau, verront aussi leur coût égal.

Notons par π_w les variables duales associées aux contraintes (3.10). La fonction objectif du sous-problème devient :

$$\min \quad 1 \quad - \sum_{w \in N} \pi_w a_{we}. \tag{3.13}$$

Les contraintes du sous-problème sont les mêmes que celles définies dans le chapitre précédent : (3.3)-(3.8). Elles déterminent une structure de chemin pour envoyer une unité de flot entre O^k et D^k. Le chemin ou le chargement sera réalisable s'il respecte les contraintes de ressource.

L'exemple suivant, qui est la suite de l'exemple de la section 2.2 du second chapitre, nous montre que, lorsque les demandes sont agrégées, la borne inférieure de la relaxation linéaire du problème se déprécie, mais elle reste toujours assez bonne.

Exemple 4 : Dans ce cas, la relaxation du problème devient le programme linéaire décrit dans le tableau 3.1 :

Tableau 3.1 – Relaxation linéaire du modèle agrégé

min	y_1	$+ y_2$	$+ y_3$		
		y_2		$= 1$	(item1)
			$2y_3$	$= 3$	(item2)
	$2y_1{}^*$	$+ y_2{}^*$		$= 2$	(item3)
	$2y_1$	$+ y_2$		$= 2$	(item4)
	$y_1,$	$y_2,$	y_3	≥ 0	

Une solution optimale est donnée par : $y_1 = \frac{1}{2}, y_2 = \frac{2}{2}, y_3 = \frac{3}{2}$, les autres $y_i = 0$ et la fonction objectif a pour valeur $\frac{6}{2} = 3$. Nous avons un nombre plus réduit

de contraintes de demande. Cependant, avec cette formulation, il est plus difficile d'éliminer la symétrie entre les items. En effet, dans le modèle non agrégé, interdire une suite d'items de même type qui ne se succèdent pas dans leur rang respectif était possible, et ces contraintes nous ont permis d'améliorer la borne inférieure. Par contre, dans le modèle agrégé, la notion de rang est éliminée et changée par celle de représentant. De ce fait, la symétrie entre ces représentants est toujours présente.

3.3 Les méthodes de résolution du problème maître

Le problème maître (3.9)-(3.12) est une version généralisée du problème de partitionnement d'ensemble dans laquelle le membre de droite de (3.10) est au moins égal à 1 et les variables de décision sont des entiers positifs. Nous recourons à l'algorithme de *"branch-and-price"* décrit dans le premier chapitre pour résoudre ce programme en nombres entiers. La borne obtenue à partir de la relaxation du problème maître est utilisée pour élaguer l'arbre de recherche et une méthode de branchement, compatible avec l'algorithme de résolution du sous-problème, doit provoquer l'émergence de la solution optimale.

Le branchement proposé dans le chapitre précédent n'est plus effectif pour ce problème général. En effet, imposer à deux items quelconques de se succéder ou non dans le même avion-cargo n'aura plus le même impact dans l'élimination des solutions fractionnaires car les items sont maintenant considérés comme des représentants et la symétrie entre eux rend le branchement moins performant. De plus, nous n'avons plus de garantie sur l'unicité d'un plan qui contient deux items successifs dans un avion, comme c'était le cas lorsque la demande pour chaque item est d'une unité. Ainsi, les plans suivants : $(item1\_4, item3\_1)$ et $(item1\_6, item3\_2)$ sont considérés comme un

seul et même plan. La décision qui impose à l'item *item*1 de ne pas succéder à l'item *item*3 élimine toute solution contenant *au moins* un de ces plans.

Dans la littérature, plusieurs auteurs résolvent des problèmes de découpe à une dimension à partir d'une formulation agrégée obtenue après décomposition ou directement comme modèle initial. Le problème est reconnu pour la borne serrée de sa relaxation linéaire. Ainsi, Marcotte (1985) montre que, dans la pratique, pour certaines classes de problèmes, l'arrondissement à l'entier supérieur de la relaxation linéaire donne assez souvent la valeur optimale de la solution entière. Scheithauer et Terno (1995) exploitent des méthodes basées sur des stratégies de réduction dans la recherche d'une solution optimale. Ils résolvent la relaxation linéaire du problème par génération de colonnes. Ils procèdent ensuite à un arrondissement inférieur de la solution obtenue pour avoir une solution partielle entière. Ainsi, le nombre de fois que chaque patron optimal est utilisé est arrondi à l'entier le précédent. Enfin, ils résolvent le problème résiduel en cherchant les patrons manquants qui couvrent la demande restante dans un *B&B* avec une formulation utilisant des variables binaires. La règle de branchement utilisée est celle qui consiste à utiliser un patron spécifique ou non. Les auteurs résolvent des problèmes à l'optimalité lorsque le nombre total de patrons trouvés dans la solution partielle entière, ajouté au nombre de patrons trouvés lors de la résolution du problème résiduel correspondant, ne dépasse pas la valeur arrondie à l'entier supérieur de la relaxation linéaire du problème initial. Nous pouvons utiliser leur méthode comme une heuristique pour obtenir rapidement une solution approchée, et ce d'autant plus que nous avons un branchement assez efficace pour résoudre le problème résiduel en variables binaires.

Les approches exactes récentes pour résoudre le problème de découpe en nombres entiers sont nombreuses. Nous avons recensé celles qui utilisent l'algorithme de génération de colonnes imbriqué dans un *B&B*. Ainsi, dans Vance *et al.* (1996), des patrons maximaux, dans le sens où aucun item supplémentaire ne peut être inséré

dans l'espace restant selon les contraintes imposées, sont générés. Le branchement s'applique sur le nombre d'utilisations d'un patron fractionnaire maximal qui satisfait une contrainte sur le nombre de composantes de chaque item figurant dans le patron. Dans une branche, le nombre d'utilisations du patron sélectionné est borné inférieurement par l'entier qui lui est directement inférieur ou égal, et dans l'autre branche, ce nombre est borné supérieurement par l'entier qui lui est directement supérieur ou égal. La première branche est traitée en ajustant le second membre des contraintes de couverture de la demande dans le problème maître. Le sous-problème n'est pas affecté par la décision. Dans la seconde branche, pour ne pas regénérer dans le sous-problème de sac de campeur les patrons bornés supérieurement, le nombre maximal d'items de certains types qui ont été découpés dans ces patrons est ajusté.

Degraève et Schrage (1999) résolvent de façon exacte des problèmes réels de découpe provenant de l'industrie et des problèmes générés aléatoirement. Leur branchement est le même que celui de Vance *et al.*, c'est-à-dire, borner supérieurement ou inférieurement un patron fractionnaire. Pour interdire la génération des patrons bornés supérieurement, ils créent une liste de colonnes interdites dans le sous-problème, et chaque fois qu'un patron prometteur est trouvé, son appartenance ou non dans la liste est vérifiée avant de l'envoyer ou non au problème maître.

Dans les méthodes citées ci-dessus, le branchement s'applique à une variable fractionnaire. Choisir d'utiliser ou non un patron ou de le borner est un schéma de branchement non balancé (Vanderbeck, 1999) dont le temps de calcul pour obtenir la solution optimale peut s'avérer, dans certains cas, très long. Cela est dû au temps pris pour faire un retour en arrière (*backtracking*) lors de l'exploration d'une branche trop restrictive et au temps pour parcourir l'autre branche dans l'espoir de trouver une solution. Ce type de branchement requiert aussi des modifications dans le sous-problème lesquelles, généralement, détruisent la structure du sous-problème (Barnhart *et al.* (1998)). Ces méthodes sont effectives si une procédure est capable

de trouver le prochain plus court chemin après chaque décision imposée dans l'arbre de recherche.

De Carvalho (1998) utilise une formulation réseau. Le problème maître contient les contraintes de couverture de la demande et les contraintes de conservation de flot. Le sous-problème est un problème de plus long chemin sur un réseau acyclique où tout chemin correspond à un patron réalisable et inversement. Le branchement est fait sur une variable de flot (arc) en le bornant supérieurement ou inférieurement, cela revient à imposer une limite sur le nombre de fois que l'item associé à cet arc est découpé sur tous les grands objets, dans le cas d'un problème de découpe. Durant la phase du *B&B*, lorsque les contraintes de branchement sont imposées dans le problème maître, des variables sont enlevées du sous-problème. Notons que les mêmes variables sont utilisées dans le problème maître et le sous-problème.

Le branchement de Vanderbeck et Wolsey (1996) est basé sur l'identification d'un ensemble de patrons qui satisfont une certaine propriété sur les composantes des items contenus dans ces patrons et dont la variable somme est fractionnaire. La séparation est effectuée en ajoutant respectivement des contraintes de borne inférieure ou supérieure dans chaque branche, sur cette variable somme. Ces contraintes sont ajoutées explicitement dans chaque problème maître restreint. Des variables auxiliaires sont définies dans le sous-problème pour tenir compte de ces modifications. Remarquons que, si l'ensemble des patrons choisis ne contient qu'un seul patron, le branchement revient à borner le patron. Les auteurs notent aussi que la règle de branchement de Ryan et Foster (1981) est une spécialisation de leur branchement lorsque la demande pour chaque item est de une unité.

Dans un autre article, Vanderbeck (2000) présente une extension du branchement de Vanderbeck et Wolsey (1996) ci-dessus. Il discute de la façon de définir une partition de l'espace des solutions et propose plusieurs critères de branchement. Dans la pro-

chaine section, nous présentons un de ces critères de branchement. Pour compléter la revue de la littérature, une autre méthode de branchement mérite d'être présentée.

A partir d'une formulation sous forme de recouvrement, le modèle de Gueguen (1999) résout par la méthode de génération de colonnes un problème réel de tournées de véhicules avec préemption de la demande (*SDVRP : Split Delivery Vehicle Routing Problem*). C'est un problème de tournées de véhicules où la demande peut être servie partiellement par plusieurs véhicules. Le modèle permet de déterminer la quantité exacte qui doit être servie chez un client par un véhicule. La génération de deux colonnes qui visitent les mêmes sommets, mais en leur servant des quantités différentes, est interdite dans l'algorithme utilisé. Pour trouver la solution entière, l'auteur branche à un premier niveau sur le nombre de véhicules utilisés et à un second niveau sur les arcs du réseau associé au sous-problème. Le branchement sur les arcs exploite la propriété décrite par Dror et Trudeau (1990) suivante : il existe une solution optimale du *SDVRP* dans laquelle aucun couple de routes ne dessert plus d'un point en commun, lorsque la matrice des coûts satisfait à la propriété de l'inégalité triangulaire. Ainsi, il en découle qu'une solution optimale dans laquelle chaque arc (i,j), où i et j sont distincts du dépôt, apparaît au plus une fois, et dès lors, les règles de branchement de Desrochers *et al.* (1992) peuvent être adaptées à leur problème. Notons que le parcours de l'arc (i,j) équivaut à effectuer la tâche associée au nœud i immédiatement avant la tâche associée au nœud j. Lorsqu'on décide que le client i est servi juste après le client j dans une branche, ou lorsqu'on interdit de servir le client j après le client i dans un autre branche, aucun arc n'est supprimé dans le sous-problème de plus court chemin avec contraintes de ressource. En effet, il est possible que dans une solution optimale, un véhicule k utilise l'arc (i,j) et qu'un autre véhicule l utilise l'arc (i,m), si les clients j et m sont différents.

Les règles de Dror et Trudeau ou celle de Gueguen ne peuvent pas être adaptées à notre modèle. L'exemple suivant permet de comprendre l'idée de leur branchement

dans le contexte de chargement d'avions. Supposons que nous ayons deux plans de chargement qui doivent charger aux points communs de demande i et j les items suivants, exemple des Jeeps : pour le premier plan, trois et deux Jeeps pour chaque point respectif i et j ; pour le second, quatre et trois respectivement pour chaque point. Il est possible qu'entre ces deux points chaque plan puisse encore charger d'autres Jeeps. Sept Jeeps doivent être chargées par les deux plans au point i, et cinq au point j. Si dès le point i le premier plan charge ses trois Jeeps et deux autres qui appartiennent normalement au second plan, et si celui-ci charge ce qui reste au point i, autrement dit, deux Jeeps et charge les cinq Jeeps au point j, les demandes sont satisfaites en ces deux points. La somme des quantités chargées par les deux plans ne change pas, et ne viole pas non plus la faisablité des deux plans. D'après la propriété de l'inégalité triangulaire, le coût de la solution ne va pas augmenter car on économise un détour. Ce raisonnement reste encore vrai pour n'importe quels deux couples d'entiers positifs (n, m) et (p, q), où chaque couple représente les quantités chargées respectivement aux deux points communs pour un plan donné, car il est toujours possible de remplacer ces couples par des nouveaux, tant que les contraintes opérationnelles ne sont pas violées. Il suffit de choisir le minimum entre ces quatre entiers comme quantité à permuter. En général, cette méthode de branchement ne peut être appliquée dans notre cas. D'abord, les permutations ne seront possibles que si les items mis en jeu sont de même type, ce qui est un cas particulier de notre problème où plusieurs types (Jeeps, palettes, camions, etc.) sont déplacés car on veut conserver à chaque permutation la faisabilité des contraintes de longueur, de poids ou de capacité. En outre, le plus souvent la contrainte du CG de l'ensemble n'est plus respectée lors de ces procédures.

3.4 Le critère de Vanderbeck

Dans cette section, nous décrivons une stratégie de branchement de Vanderbeck (2000). Nous avons choisi de présenter celle où le sous-problème de la génération de colonnes peut être formulé comme un problème de plus court chemin dans un réseau acyclique et est résolu par programmation dynamique. Cette méthode a été validée sur des problèmes de découpe et l'auteur a procédé à des tests de comparaison avec la méthode de Vance *et al.* (1996) et celle de de Carvalho (1998). Les résultats ont mis en évidence l'efficacité de sa méthode.

Lorsque chaque colonne générée par le sous-problème peut être associée à un chemin dans un réseau, Vanderbeck propose un branchement particulier. Une solution de la relaxation linéaire (3.9)-(3.11) correspond à une combinaison de chemins satisfaisant les contraintes liantes. Ainsi, un item particulier pourra apparaître dans plusieurs chemins à différentes positions, et c'est la somme des variables associées à ces chemins qui sera entière. Le branchement consiste à selectionner un arc dans le réseau, et d'imposer que la somme des variables associées aux chemins utilisant cet arc soit entière.

Soit donc un arc (i, j) quelconque dans le réseau, et soit X_{ij} la somme des flots sur tous les avions utilisant cet arc :

$$X_{ij} = \sum_{k \in K:(i,j) \in A^k} X_{ij}^k, \quad \forall (i,j) \in \bigcup_{k \in K} A^k.$$

Comme les plans de chargement sont utilisés un nombre entier de fois, il faut que les variables de flot X_{ij}, associées aux parcours des arcs (i, j) pour former ces plans, soient aussi entières. Dans le cas contraire, une variable X_{ij} de valeur fractionnaire f_{ij} devient, de ce fait, candidate au branchement. Les décisions possibles sont :

$$X_{ij} \leq \lfloor f_{ij} \rfloor \tag{3.14}$$

sur une branche et

$$X_{ij} \geq \lceil f_{ij} \rceil \tag{3.15}$$

sur une autre. La validation de cette division découle du théorème sur la décomposition du flot, voir Ahuja *et al.* (1993). Ainsi, si le réseau est acyclique et si le flot sur chaque arc du réseau sous-jacent est entier, ce flot entier, grâce au théorème de la décomposition du flot, peut être décomposé en flots entiers le long d'un nombre fini de chemins où chacun d'eux correspond à une solution du problème. Donc, une solution entière est trouvée. Sinon, si la solution est non entière, alors il existe un arc du réseau pour lequel le flot sur cet arc est fractionnaire, et toute solution fractionnaire peut être éliminée en utilisant ces contraintes de branchement.

Le problème maître va tenir compte de ces contraintes si nous exprimons ces dernières en termes des variables de chemin θ_e ($e \in \Omega$) de la formulation de la génération de colonnes, c'est-à-dire $X_{ij} = \sum_{e \in \Omega} x_{ije}\theta_e$ où $x_{ije} = 1$ si l'arc (i, j) est dans le chemin e et 0 sinon.

A un nœud u quelconque de l'arbre de recherche, la formulation du problème maître relaxé s'écrit :

$$\min \sum_{e \in \Omega} \theta_e \tag{3.16}$$

sujet aux contraintes :

$$\sum_{e \in \Omega} a_{we}\theta_e = q_w, \forall w \in N \tag{3.17}$$

$$\sum_{e \in \Omega} x_{ije}\theta_e \leq \lfloor f_{ij} \rfloor, \forall (i,j) \in LO^u \tag{3.18}$$

$$\sum_{e \in \Omega} x_{ije}\theta_e \geq \lceil f_{ij} \rceil, \forall (i,j) \in UP^u \tag{3.19}$$

$$\theta_e \geq 0, \forall e \in \Omega \tag{3.20}$$

où LO^u et UP^u désignent, respectivement, l'ensemble des décisions de branchement du type (3.14) et du type (3.15). Ces ensembles définissent le problème au nœud u de l'arbre de recherche d'un $B\&B$.

Rappelons que la fonction objectif du sous-problème k est la somme des valeurs marginales des items inclus dans le plan correspondant. Les modifications sur cette fonction résultant des contraintes de branchement à un nœud u de l'arbre de recherche prennent la forme suivante :

$$1 - \sum_{(i,j)\in A} (\pi_i + \alpha_{ij} + \beta_{ij}) X_{ij}^k \tag{3.21}$$

dans lesquelles nous posons $A = \bigcup_{k\in K} A^k$, $\alpha_{ij} \leq 0 \quad ((i,j) \in LO^u)$ les variables duales associées aux contraintes (3.18) et $\beta_{ij} \geq 0 \quad ((i,j) \in UP^u)$ les variables duales associées aux contraintes (3.19). Par convention, nous posons $\alpha_{ij} = 0$ si $(i,j) \notin LO^u$ et $\beta_{ij} = 0$ si $(i,j) \notin UP^u$. La structure du sous-problème reste la même et l'effet des décisions de branchement se traduit par la modification des coûts des arcs impliqués.

Notons que dans son article, Vanderbeck (2000) mentionne que le branchement de De Carvalho (1998) utilise des règles similaires et que les modifications effectuées dans le sous-problème sont simples mais que le branchement, proposé par ce dernier, peut ne pas être efficace. La sélection de la variable de branchement (arc) de la méthode de De Carvalho n'est pas facile car la variable est choisie parmi un grand nombre d'arcs potentiels et que le branchement sur un seul arc n'est pas tellement fort. De plus, cette méthode de branchement doit être modifiée pour résoudre des problèmes formulés sous forme de partitionnement où le second membre est de 1 unité. Dans ce cas, Vanderbeck propose l'identification d'un ensemble de patrons qui passent par deux arcs bien précis du réseau et dont la variable somme est fractionnaire.

Vanderbeck (2000) note aussi que lorsque le sous-problème de la méthode de génération de colonnes est un problème de plus court chemin avec contraintes de ressource, le théorème de la décomposition de flot ne s'applique plus sur le réseau associé au sous-problème résolu par le plus court chemin avec contraintes de ressource. La validation de sa règle de branchement est effective seulement sur un réseau associé avec

un sous-problème résolu par programmation dynamique. Le contre-exemple suivant décrit une instance où l'algorithme n'arrive pas à trouver un arc sur quoi brancher pour éliminer la solution fractionnaire.

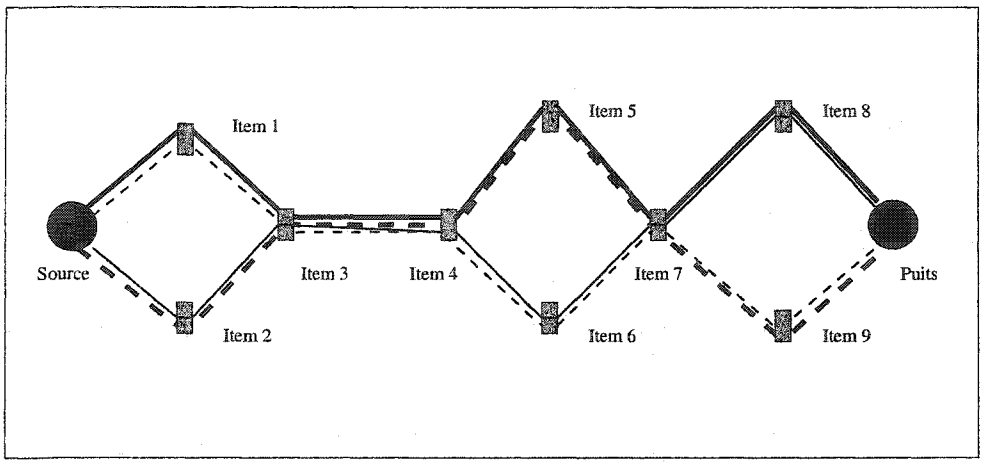

Figure 3.2 – Une solution où chaque trait représente un chemin

Exemple : La Figure 3.2 montre le réseau associé à une instance de neuf items à charger, répartis comme suit : $q_w = 1, w \in \{1, 2, 5, 6, 8, 9\}$ et $q_w = 2, w \in \{3, 4, 7\}$. Supposons que chaque plan de chargement $\theta_e = \frac{1}{2}, e \in \{1, 2, 3, 4\}$ correspond à un chemin réalisable, donc, chacun satisfait aux contraintes de ressource. Le coût de cette solution fractionnaire est de deux unités et le flot sur chaque arc du réseau est entier. Ainsi, nous ne pouvons pas appliquer les règles de branchement de Vanderbeck. De plus, nous ne pouvons pas construire une solution entière à partir de ces quatre plans au risque de ne pas satisfaire les contraintes de ressource.

La section suivante présente les résultats des tests numériques effectués avec cette approche heuristique. La même coupe que dans le second chapitre est utilisée pour renforcer la relaxation linéaire à travers l'arbre de recherche. Pour résoudre les instances présentées, l'arbre de recherche du *B&B* est exploré en utilisant aussi la même procédure que dans le second chapitre.

3.5 Résultats numériques avec l'approche heuristique

Nous avons testé le critère de Vanderbeck sur le problème réel de Ng (1992) et ses dérivés, et les problèmes générés aléatoirement du chapitre précédent. Pour l'ensemble des tests effectués, la variable de flot dont la partie fractionnaire est la plus près de 1 est choisie comme variable de branchement. Ce choix permet de trouver rapidement une solution entière, du fait que nous changeons de branche dès que la décision implique un avion en plus. Pour chaque problème test, nous mentionnons, dans chaque tableau de résultats, le nombre de priorités d'envoi des items (*Priorité*), le temps de résolution agrégé (*CPUC*) en secondes, le temps de résolution non agrégé de l'algorithme du chapitre précédent (*CPUA*) en secondes, la solution optimale (*Sol*), la valeur de la relaxation linéaire (*Relax*), le nombre de nœuds explorés (*B&B*), le nombre de colonnes générées (*ColGenC*) dans le cas agrégé et le nombre de colonnes générées (*ColGenA*) dans le cas non agrégé, le nombre d'arcs et de nœuds pour le nouveau réseau (*ArcsC*, *NœudsC*) et pour le réseau du modèle désagrégé (*ArcsA*, *NœudsA*), la mémoire requise pour résoudre le problème dans le cas agrégé (*MemC*) et dans le cas désagrégé (*MemA*) en mégaoctets.

Les résultats obtenus à partir des données de Ng et des instances dérivées sont présentés dans les tableaux 3.2, 3.3 et 3.4 qui décrivent respectivement les résultats pour différentes largeurs de l'intervalle de l'admissibilité du CG. La définition de chaque colonne "*Une*" (resp. "*Deux*, *Trois*, *Quatre*") est toujours la même que dans le chapitre précédent ; la première est le problème de Ng, la seconde est le double de la taille du problème de Ng et ainsi de suite. Notons que pour chacun de ces problèmes et dans chaque réseau, six séries sont fixées sur le quai et un sur la rampe.

Le premier constat est qu'il est possible de résoudre à l'optimalité le problème réel de Ng en moins de 2 secondes avec quatre priorités ou aucune priorité fixée sur l'envoi

Tableau 3.2 – Intervalle large pour le CG : (245,737)

Priorité	1 Priorité				4 Priorités			
modèle	Une	Deux	Trois	Quatre	Une	Deux	Trois	Quatre
CPUC	1	1	2	1	1	1	1	1
CPUA	202	1963	8715	22590	77	655	2963	8646
Sol	92	184	275	367	101	201	302	402
Relax	91.6	183.1	274.6	366.2	100.5	201.0	301.5	402.0
B&B	38	34	53	48	35	1	24	1
ColGenC	167	124	211	157	158	110	119	110
ColGenA	4293	9356	14150	19316	3026	6422	9515	13039
ArcsC	2668	2680	2680	2680	2703	2727	2727	2727
ArcsA	8159	14273	20387	26501	10476	19266	28056	36846
NœudsC	439	445	445	445	439	445	445	445
NœudsA	2291	4329	6367	8405	2291	4329	6367	8405
MemC	8.1	8.1	8.1	8.1	1.5	1.5	1.5	1.5
MemA	76.4	155.0	241.0	334.3	11.0	26.1	46.2	70.7

des items. De plus, il suffit de quelques centaines de plans pour trouver la solution de chaque problème comparé à quelques milliers dans le modèle non agrégé. En effet, un plan de chargement généré par cette formulation peut être utilisé de façon combinatoire autant de fois que les items contenus dans le plan le permettent. Enfin, la nouvelle formulation est moins sensible lorsque nous multiplions par deux, trois ou quatre fois les exemplaires de chaque item à transporter. Ainsi, la difficulté de chaque problème, pour une priorité fixée, semble être la même. Comme le nombre d'items à transporter reste constant, le nombre de rangées ou contraintes dans le problème maître reste aussi constant, mais c'est le nombre de demandes qui croît. Les nombres d'arcs et de nœuds pour décrire chaque problème sont à peu près les mêmes car leurs valeurs respectives maximales sont atteintes lorsque la taille du problème est au moins le double de celle de Ng. Pour cette raison, dans le processus de l'algorithme de la programmation dynamique, la mémoire utilisée et le temps CPU sont relativement semblables. Par contre, avec le modèle non agrégé du chapitre précédent, chaque copie est spécifique et chacune correspond à une rangée du problème maître. Dans ce cas, la difficulté de chaque problème croît avec le nombre de copies spécifiques transportées,

Tableau 3.3 – Intervalle moyen pour le CG : (551,564)

Priorité	1 Priorité				4 Priorités			
modèle	Une	Deux	Trois	Quatre	Une	Deux	Trois	Quatre
CPUC	2	2	2	2	1	1	1	1
CPUA	319	2410	9589	23680	83	805	3588	8335
Sol	92	184	276	368	106	212	318	424
Relax	92.0	184.0	276.0	368.0	106.0	212.0	318.0	424.0
B&B	1	1	1	1	38	13	10	1
ColGenC	141	141	141	141	156	178	166	138
ColGenA	9383	26127	56548	59151	3077	6771	9828	13899
ArcsC	2668	2680	2680	2680	2703	2727	2727	2727
ArcsA	8159	14273	20387	26501	10476	19266	28056	36846
NœudsC	439	445	445	445	439	445	445	445
NœudsA	2291	4329	6367	8405	2291	4329	6367	8405
MemC	5.1	5.2	5.2	5.2	1.4	1.4	1.4	1.4
MemA	57.8	119.7	189.2	263.2	10.7	25.9	45.9	76.2

Tableau 3.4 – Intervalle serré pour le CG : (559,563)

Priorité	1 Priorité				4 Priorités			
modèle	Une	Deux	Trois	Quatre	Une	Deux	Trois	Quatre
CPUC	1	2	2	3	1	1	1	1
CPUA	216	1503	5860	12770	93	869	3776	9747
Sol	97	194	291	388	108	216	324	432
Relax	96.9	193.7	290.6	387.4	108.0	216.0	324.0	432.0
B&B	18	25	30	49	25	1	23	1
ColGenC	124	138	171	373	140	121	123	121
ColGenA	7567	17863	33949	46319	3276	6735	10002	13665
ArcsC	2668	2680	2680	2680	2703	2727	2727	2727
ArcsA	8159	14273	20387	26501	10476	19266	28056	36846
NœudsC	439	445	445	445	439	445	445	445
NœudsA	2291	4329	6367	8405	2291	4329	6367	8405
MemC	4.0	4.2	4.2	4.2	1.4	1.4	1.4	1.4
MemA	46.7	98.0	157.7	223.0	10.7	25.2	44.0	68.4

même si les autres paramètres du problème, tels le nombre de colonnes générées, le nombre d'arcs ou de nœuds, etc. augmentent de façon linéaire. La résolution du sous-problème influence le temps de calcul à cause du nombre d'étiquettes qui varient en fonction du nombre et de la longueur des items transportés.

Pour les problèmes générés aléatoirement, les résultats sont présentés dans les tableaux 3.5 et 3.6. Ces problèmes sont les mêmes que ceux du chapitre précédent. Rappelons que le nombre d'items pour chaque problème est de 20 unités et que la taille de chaque est toujours plus grande que celle du problème réel de Ng. L'intervalle du CG est fixé à (551,564) et aucune priorité n'est imposée. La moyenne et l'écart type sont donnés dans les deux premières colonnes : MOY et $E.T.$

Les résultats montrent que l'algorithme résout chaque problème dans un temps très court comparé à la méthode du chapitre précédent. L'algorithme résout les problèmes en moyenne 140 fois plus vite que la méthode non agrégée et le nombre de nœuds de branchement est moins d'une centaine. Nous avons aussi des gains considérables en termes de taille du réseau et de mémoire (en moyenne 15 fois moins) utilisée pour chaque problème résolu. Le nombre de colonnes générées est de l'ordre de quelques centaines comparé aux milliers de la méthode non agrégée. Une des justifications de la résolution de tous ces problèmes à l'optimalité serait que la borne linéaire est très proche de la valeur de la solution optimale. En effet, le saut entre la valeur de la relaxation linéaire et la solution optimale est moins d'une unité. Il a été possible de résoudre tous les problèmes à l'optimalité. Ces résultats montrent l'efficacité de la méthode de branchement pour résoudre la classe de problème et mieux comprendre les symétries sur les items.

Ainsi, les résultats numériques de cette section sont excellents, mais non garantis. Dans la section suivante, nous développons et testons un algorithme exact qui s'adapte avec des problèmes formulés sous forme de partitionnement d'ensemble où le second membre est un nombre entier quelconque plus grand ou égal à 1.

Tableau 3.5 – Problèmes aléatoires (partie 1)

	MOY	E.T	N1	N2	N3	N4	N5	N6
CPUC	11	16	2	1	4	41	24	1
CPUA	1628	2015	340	224	763	4748	2197	552
Sol	145	26	135	120	151	103	151	140
Relax	144.2	25.7	134.8	119.4	151.0	102.6	150.8	139.3
B&B	47	14	37	60	14	50	69	40
ColGenC	242	140	111	142	145	422	708	125
ColGenA	9959	7873	4261	4237	6681	13511	9025	5445
ArcsC	2706	479	2220	2211	2297	2884	2833	2213
ArcsA	11059	2281	8249	7460	9331	10922	13048	9303
NœudsC	487	97	394	388	432	547	521	391
NœudsA	3272	698	2411	2143	2770	3212	3919	2764
MemC	17.8	22.4	5.6	1.6	14.2	62.5	17.2	3.8
MemA	270.8	346.4	59.2	19.0	192.0	913.8	291.7	55.2
	N7	N8	N9	N10	N11	N12	N13	N14
CPUC	5	14	2	7	7	14	1	2
CPUA	510	1282	1145	704	1614	1949	227	1066
Sol	157	146	200	133	195	120	131	146
Relax	156.9	145.9	199.8	132.5	194.3	119.4	131.0	145.6
B&B	39	22	29	53	64	41	33	46
ColGenC	220	124	119	185	284	201	135	177
ColGenA	5475	5549	7092	6277	10722	9790	3976	6972
ArcsC	2734	3919	2247	2872	3326	2788	2186	2701
ArcsA	10686	14035	11407	10858	15527	10183	8089	12073
NœudsC	471	703	407	539	586	497	378	455
NœudsA	3130	4058	3462	3184	4647	2959	2361	3594
MemC	13.8	35.8	5.5	12.7	11.9	19.2	4.7	5.2
MemA	165.7	358.1	95.6	223.1	243.7	417.1	48.9	89.2
	N15	N16	N17	N18	N19	N20	N21	N22
CPUC	6	5	4	25	19	1	6	7
CPUA	1144	823	1394	4370	1212	458	350	892
Sol	137	112	152	124	111	162	159	118
Relax	136.5	112.0	151.2	123.5	110.8	161.8	158.4	117.5
B&B	65	30	53	54	32	45	57	57
ColGenC	240	216	251	375	296	218	291	269
ColGenA	9523	19645	9934	30464	11951	4867	4329	10027
ArcsC	3396	2224	2773	2333	2893	2150	2746	2283
ArcsA	13094	8379	12667	10692	10726	9461	10848	8940
NœudsC	621	395	491	451	551	359	477	426
NœudsA	3836	2451	3792	3227	3145	2815	3184	2643
MemC	12.6	5.4	9.0	38.8	30.1	2.1	6.4	16.2
MemA	191.9	113.4	167.0	752.0	273.6	32.2	57.5	196.7

Tableau 3.6 – Problèmes aléatoires (partie 2)

	N23	N24	N25	N26	N27	N28	N29	N30
CPUC	1	79	8	6	2	6	38	3
CPA	175	9782	1021	338	472	1581	5781	1736
Sol	188	114	154	128	174	166	123	186
Relax	188.0	113.9	154.0	127.4	173.2	165.5	122.9	185.3
B&B	33	37	54	63	52	66	69	46
ColGenC	136	288	177	141	161	259	637	197
ColGenA	3142	27489	8068	4177	5209	7089	35321	8516
ArcsC	2087	3517	3329	2743	2210	3390	2905	2764
ArcsA	8821	14471	13017	8966	9917	15556	13027	14048
NœudsC	327	683	588	475	389	619	557	487
NœudsA	2600	4300	3812	2555	2967	4660	3912	4254
MemC	1.5	111.2	11.6	11.7	3.8	16.1	37.9	6.0
MemA	22.1	1708.1	145.5	81.1	39.0	332.7	740.3	97.4

3.6 Une stratégie de branchement

Dans cette section, nous présentons une stratégie de branchement qui peut être adaptée à la formulation (3.1)-(3.8). Le sous-problème est un problème de plus court chemin avec contraintes de ressource et est résolu par programmation dynamique. Considérons le problème de partitionnement d'ensemble où les membres de droite des contraintes de chargement (2.2) est d'une unité. Le branchement développé par Desrochers et Soumis (1989) consiste, dans ce cas, à identifier selon un critère bien précis deux items spécifiques et susceptibles d'être chargés successivement. L'espace de solutions est divisé en deux parties. Celle où le chargement contenant le premier de ces items contient l'autre chargé immédiatement après. L'autre partie contient les solutions où ces items ne sont jamais chargés le second à la suite du premier. Cette règle élimine la solution fractionnaire courante sans compromettre l'accessibilité des solutions dans le sous-problème. Lorsque le membre de droite d'une contrainte de chargement (3.2) est plus grand ou égal à une unité, c'est-à-dire dans le cas agrégé, ce branchement ne peut s'appliquer puisque les items ne sont plus "individualisés".

Une solution optimale pourra être constituée de chargements où deux types d'items seront "en succession" dans certains chargements et non dans les autres. Nous proposons une extension du branchement de Desrochers et Soumis. Elle est encore basée sur les variables de flot entre deux tâches consécutives. Cette nouvelle proposition est innovatrice en deux points. On ne considère plus si deux items particuliers doivent être chargés successivement, mais on considère plutôt si deux types d'items seront chargés successivement. De plus, elle consiste à construire un par un les plans de chargement. De ce fait, la sélection de la prochaine variable de branchement se fait en fonction du prolongement d'un plan de chargement déjà commencé.

Nous commençons par décrire le branchement dans le cas où la demande de chargement, pour chaque item, serait d'une unité. Pour réduire l'effet des symétries entre les items ou les plans de chargement, de nouvelles règles sont proposées. Ensuite, nous montrons comment le branchement peut être adapté à un problème où la demande de chargement, pour chaque item, est un nombre entier plus grand ou égal à une unité. Des expériences numériques terminent la section et montrent le comportement de l'algorithme sur les problèmes tests.

3.6.1 Cas non agrégé

Pour mieux comprendre notre procédure de branchement, revenons sur le problème de partitionnement d'ensemble où le membre de droite des contraintes de chargement (2.2) est d'une unité. Nous proposons, pour obtenir une solution optimale d'un problème de chargement, l'algorithme suivant :

Etape 0 : (Initialisation)

- Créer une liste L des items ordonnée du plus long au plus court ;
- Résoudre la relaxation linéaire, aller à l'étape 1.

Etape 1 : (Test d'intégralité de la solution)

- Si la solution est entière alors stop ;
- sinon aller à l'étape 2.

Etape 2 : (Ouverture d'une séquence)

- Traiter le premier item de la liste L ;
- Sélectionner un autre item.

Etape 3 : (Prolongement d'une séquence de chargement)

- Modifier les contraintes de branchement ;
- Fixer (ou défaire) les items en succession ;
- Mettre à jour la liste L ;
- Résoudre la relaxation linéaire.

Etape 4 : (Test d'intégralité de la solution)

- Si la solution est entière alors
  1. si le critère d'arrêt est atteint, alors stop ;
  2. sinon aller à l'étape 3 (*backtracking*).
- Sinon aller à l'étape 5.

Etape 5 : (Test de complétion d'un chargement)

- Si le chargement est complété, aller à l'étape 2 ;
- sinon aller à l'étape 3.

L'étape 0 effectue le prétraitement des données et le calcul de la solution initiale du problème relaxé. Remarquons que la liste L est ordonnée selon la longueur des items car c'est la contrainte la plus limitative avec les données que nous avons. En

choisissant l'item le plus long, nous espérons réduire la combinatoire pour construire un chargement complet.

L'étape 1 vérifie l'intégrité de la solution. Lorsque la solution est entière, nous avons la solution optimale.

Sinon, à l'étape 2 nous commençons à choisir l'item le plus long de la liste et à chercher par la suite l'item qui peut le précèder ou celui qui peut le succèder dans un chargement. Le critère de sélection est basé sur la solution de la relaxation linéaire. Nous avons opté pour une valeur du flot, entre l'item candidat et l'item traité, qui est la plus proche de 1. En cas de litige entre deux items candidats, le plus long est choisi.

L'étape 3 procède au prolongement de la séquence courante. Les décisions sur les items à charger ou à ne pas charger par la suite sont prises à ce niveau et une mise à jour des items restants est effectuée, suivie de la résolution de la relaxation linéaire du nouveau problème. Sans perte de généralité, nous choisissons l'item candidat et l'item traité qui commencent une séquence comme suit. Lorsque les deux items sont de types différents, l'item qui doit être placé à l'avant (le plus près de la tête de l'avion) est substitué par un item de même type encore disponible et de rang maximal. L'item chargé immédiatement après est choisi parmi les items de même type encore disponible et de rang minimal. Cette procédure favorise le chargement d'un nombre maximum d'items respectivement à l'avant et à l'arrière du couple choisi. Rappelons que deux items de même type ne peuvent se succéder que si leur rang respectif sont consécutifs. De même, si l'item candidat est placé à l'avant (resp. à l'arrière) d'une séquence d'items déjà chargés, nous le substituons avec un item encore disponible de même type et de rang maximal (resp. minimal). Par ailleurs, lorsque l'item candidat et l'item traité sont de même type, aucune substitution n'est effectuée. Leur rang respectif se succèdent déjà dans le réseau.

L'étape 4 de l'algorithme vérifie, encore une fois, l'intégralité de la solution. Lorsque celle-ci est entière, nous avons une borne supérieure. Selon un critère d'arrêt prédéfini, la solution peut être satisfaisante. Sinon, un retour en arrière s'effectue (*backtracking*) et on défait la séquence courante. Ainsi, à l'étape 3 et lors de la création d'un chargement, lorsque les deux types ne sont pas compatibles c'est-à-dire si la décision de chargement interdit la succession du couple d'items, toutes les connexions entre ces deux types ne sont pas autorisées dans tout le réseau. En particulier, lorsque les deux items sont de même type, toutes les successions entre ces items sont interdites. De la même façon, lorsqu'une décision quelconque interdit de prolonger une séquence formée au moins de trois items, par l'avant ou par l'arrière selon le cas, avec un item de type donné, ce sont tous les items du type qui sont impliqués dans l'interdiction. Cela nous évite de chercher des candidats non prometteurs.

L'étape 5 procède à un test de complétion du chargement courant. Si la complétion est réalisée, un autre chargement doit être formé et l'étape 2 s'en suit. Si le plan n'est pas complété, le prolongement de la séquence continue et nous retournons à l'étape 3. Notons qu'un chargement est complété si les nœuds *source* et *puits* sont présents dans le chemin. Par ailleurs, si le flot entre les nœuds *source* et *Item* ou entre les nœuds *Item* et *puits*, auquel cas *Item* désigne un nœud associé à un item à charger, est différent de 1, une décision doit être prise sur l'extension ou non de la séquence courante vers la *source* ou le *puits*, selon le cas.

Illustrons la méthode à l'aide de l'exemple suivant. Soit les trois plans *(Item3_2, Item8_2, Item1_1, Item15_1, Item1_7)*, *(Item8_2, Item1_1, Item15_1, Item4_4)* et *(Item3_5, Item8_2, Item1_1, Item15_1, Item1_4)*. Ces trois plans contiennent tous le chargement partiel *(Item8_2, Item1_1, Item15_1)*. Pour forcer (resp. empêcher) une séquence de s'étendre d'un item déjà chargé vers un item particulier, nous fixons (resp. interdisons) la succession entre l'item déjà chargé de la séquence et l'item particulier. Ainsi, si l'exemplaire *Item1_7* est choisi selon un critère prédeterminé

pour succéder à l'exemplaire *Item15_1* dans la séquence, c'est finalement l'exemplaire *Item1_2* qui sera mis dans la suite car, après substitution, c'est le prochain dans la liste qui ne soit pas encore fixé pour le type *Item1*. Les trois plans n'apparaîtront plus dans l'ensemble des solutions issu de cette décision, mais un ou plusieurs autres plans contenant la nouvelle séquence *(Item8_2, Item1_1, Item15_1, Item1_2)* seront générés. Dans l'autre branche, nous interdisons tous les représentants de *Item1* à succéder l'item *Item15_1*. Parmi les trois plans, seul le second peut faire partie de l'ensemble des solutions issu de cette décision.

Remarquons que, les règles de branchement sont plus générales que la méthode standard car elles affectent les types d'items plutôt que les items. Fixer deux types à se succéder est une décision assez forte car c'est toujours deux exemplaires spécifiques qui seront fixés. Interdire deux types à se succéder est aussi une décision assez forte car elle implique plusieurs plans à la fois. Ce mode de branchement permet aussi d'équilibrer l'arbre de recherche car plusieurs plans de chargement sont évalués à chaque décision. De plus, même si le branchement s'applique sur les types et non sur les items spécifiques, l'arbre de recherche a la même profondeur que celui de Desrochers et Soumis (1989). L'exploration de cet arbre est différente car nous imposons une priorité sur les branches à parcourir, en primant la complétion d'un chargement plutôt que la création d'un autre.

Remarquons aussi que, chaque fois qu'un exemplaire est fixé, il n'est plus candidat au branchement. En particulier, une fois qu'un plan de chargement est trouvé, implicitement un réseau résiduel apparaît et les membres de droite des contraintes de demande de chargement peuvent être mis à jour. Dans ce cas, pour chaque type d'items, c'est l'item portant le dernier rang dans le type qui est éliminé en premier et ainsi de suite. Pour éviter de résoudre le même problème résiduel, le critère suivant est utilisé pour dominer certains sous-arbres de recherche non prometteurs.

Proposition 4 *: Soit N_1 (resp. N_2) le rang d'un nœud de l'arbre de recherche où la borne inférieure du problème résiduel est noté par B_1 (resp. B_2), le nombre de plans fixés est noté par S_1 (resp. S_2) et le vecteur à n composantes formé par la demande restante est noté par $\mathbf{V}_1$ (resp. $\mathbf{V}_2$). Si le nœud N_1 a déjà été exploré, et que les conditions suivantes sont vérifiées au nœud N_2 : $B_1 \leq B_2$, $S_1 = S_2$,$\mathbf{V}_1 = \mathbf{V}_2$ alors il est inutile d'explorer l'arborescence enracinée au nœud N_2.*

Preuve : La preuve est évidente. □

Nous avons l'égalité sur le nombre des vecteurs de demande car le contre-exemple suivant illustre le fait que le critère ne fonctionne plus, dans certains cas, à cause de la contrainte du CG.

Contre-exemple : Supposons que nous ayons à transporter les deux types d'items suivants : $Item1$ de longueur 3 unités et de poids 1 unité et $Item2$ de longueur 1 unité et de poids 3 unités. Nous supposons que le CG de chaque item est situé en son centre. En outre, l'avion a un quai de 5 unités de longueur et est capable de supporter jusqu'à 8 unités de poids. Soit les deux instances I_1 et I_2 suivantes de demandes respectives :

- I_1 : 1 $Item1$ et 1 $Item2$.
- I_2 : 1 $Item1$ et 2 $Item2$.

Le CG de l'ensemble avion-cargo doit être situé au milieu de l'avion, dans un rayon de tolérance très petit, de sorte que pour la première instance, deux avions sont requis pour le transport de ces deux items. Par contre, un seul avion est suffisant pour transporter les trois items de la seconde instance en plaçant le long item entre les deux autres. A la lumière de ce contre-exemple, lorsque le nombre d'items à charger augmente le nombre d'avions requis pour les transporter peut être réduit.

3.6.2 Cas agrégé

Nous montrons comment adapter les règles de branchement de la section précédente au cas agrégé. L'idée générale de la méthode est de travailler avec une formulation agrégée en utilisant des règles de branchement fonctionnant sur des modèles non agrégés. Pour chaque type d'item, la contrainte de couverture est partitionnée en un nombre minimum de contraintes de demande d'une unité et, éventuellement, une autre contrainte de demande plus grande que l'unité. Les règles de branchement précédentes sont appliquées à une partie du réseau original sans pour autant perdre des solutions potentielles. Nous espérons, dans ce cas, réduire le temps de calcul et la mémoire consommée lors de la résolution du problème, moyennant une perte de la qualité de la borne inférieure. Nous présentons les changements opérés dans le réseau et nous précisons qu'en fait, nous avons une généralisation du réseau présenté dans le second chapitre. Un exemple est donné pour faciliter la compréhension du réseau et les modifications effectuées.

Pour le type d'items $w, w \in \{1, 2, ..., |N|\}$, on divise les q_w items en deux blocs. Dans le premier bloc, on mettra le nombre maximal d'items pouvant être chargés dans un avion. Ce nombre correspond à $\min\{r_w + p_w, q_w\}$. Rappelons que $r_w = \min\{\lfloor \frac{L_B}{l_w} \rfloor, \lfloor \frac{M_B}{m_w} \rfloor, q_w\}$ (respectivement $p_w = \min\{\lfloor \frac{L_R}{l_w} \rfloor, \lfloor \frac{M_R}{m_w} \rfloor, q_w\}$) est le nombre maximal d'exemplaires pour le groupe w pouvant être chargés sur le quai (respectivement sur la rampe). Ces items seront traités comme non agrégés, c'est-à-dire qu'ils auront une demande de 1 unité dans les contraintes de demande. Le deuxième bloc contient tous les autres items de type w. Ces items seront traités comme agrégés. Le membre de droite des contraintes de demande associées à ces items est égal à $q'_w = \max\{q_w - r_w - p_w, 0\}$ au nœud racine de l'arbre d'énumération et sera mis à jour au fur et à mesure qu'on évoluera dans cet arbre.

Les noeuds du réseau sont encore groupés en séries et sont situés en diverses positions dans l'avion. Les changements dans le réseau se situent essentiellement à l'intérieur

de chaque groupe d'items. Pour chaque groupe $w, w \in \{1, ..., |N^k|\}$, associé à l'avion k, un nœud $I^k_{w\, z_w\, s}$ est associé à chaque exemplaire $z_w, z_w = 1, ..., \min\{r_w + p_w, q_w\}$, pour chaque série $s \in S$ sur le quai et éventuellement sur la rampe. Des nœuds $I'^k_{w\, z_w\, s}$ ($z_w = 1, ..., r'_w$ où $r'_w = \min\{\lfloor \frac{L_B}{l_w} \rfloor, \lfloor \frac{W_B}{m_w} \rfloor, q'_w\}$) représentent les exemplaires de l'item w du bloc agrégé, pour chaque série $s \in S$ sur le quai. De même, pour chaque série $s \in S$ sur la rampe, des nœuds $I'^k_{w\, z_w\, s}$ ($z_w = 1, ..., p'_w$ où $p'_w = \min\{\lfloor \frac{L_R}{l_w} \rfloor, \lfloor \frac{W_R}{m_w} \rfloor, q'_w\}$) repésentent les exemplaires de l'item w du bloc agrégé, si l'item peut être mis sur la rampe.

L'ensemble d'arcs A^k subit uniquement des modifications à l'intérieur d'un groupe d'items (voir Figure 3.3). Ainsi, les arcs du type 1,5,6 de la section 2.3 du second chapitre sont gardés. Par contre, ceux du type 2, 3, 4 situés au sein d'un groupe sont modifiés. On obtient :

1. Les arcs $(B^k_{ws}, I^k_{w\, z_w\, s}), w = 1, ..., |N^k|, z_w = 1, ..., r_w + p_w$ sont définis pour permettre la sélection d'un exemplaire de l'item w qui sera chargé en premier dans la série s, pour le bloc non agrégé.
2. Les arcs $(B^k_{ws}, I'^k_{w1s}), w = 1, ..., |N^k|$ sont définis pour permettre la sélection d'un exemplaire de l'item w de la série s du bloc agrégé et qui sera chargé dans la suite du processus.
3. Les arcs $(I^k_{w\, z_w\, s}, I^k_{w\, z_w+1\, s}), z_w = 1, ..., r_w + p_w - 1$, sont définis pour permettre à deux exemplaires de même type w, de la série s, du bloc non agrégé, d'être mis l'un après l'autre sur le quai ou, éventuellement, sur la rampe.
4. Les arcs $(I^k_{w\, z_w\, s}, I'^k_{w1s}), z_w = r_w + p_w$ sont définis pour permettre à deux exemplaires, l'un du bloc non agrégé et l'autre agrégé, de même type w et de la série s, d'être mis l'un après l'autre sur le quai ou, éventuellement, sur la rampe.
5. Les arcs $(I'^k_{w\, z_w\, s}, I'^k_{w\, z_w+1\, s}), z_w = 1, ..., r'_w - 1$ ou $z_w = 1, ..., p'_w - 1$, sont définis pour permettre à deux exemplaires de même type w et de la série s, du bloc agrégé, d'être mis l'un après l'autre sur le quai ou, éventuellement, sur la rampe.

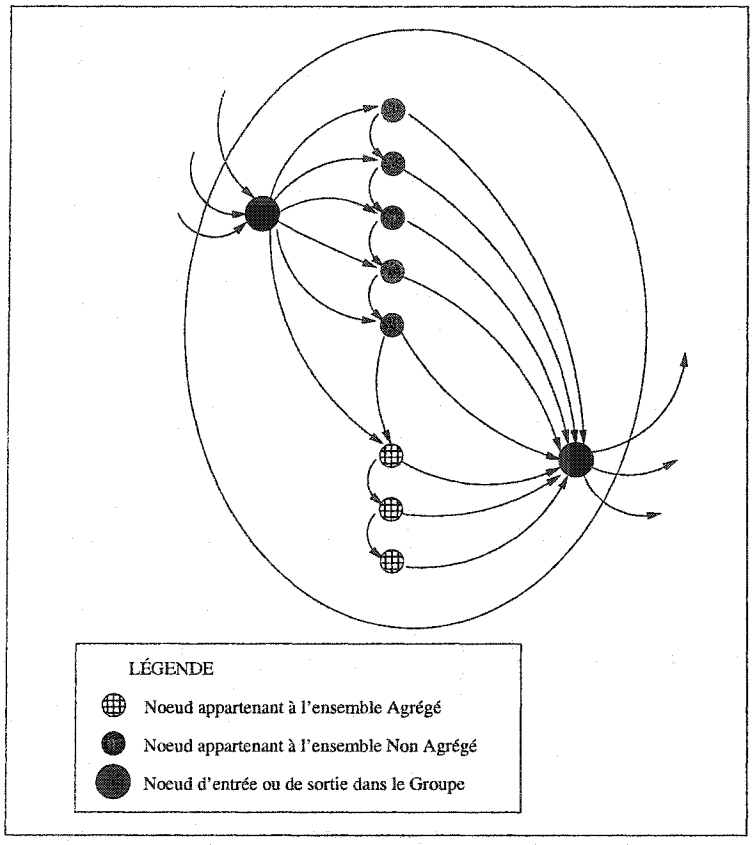

Figure 3.3 – Groupe d'items de même type

6. Les arcs $(I^k_{w\,z_w\,s}, E^k_{ws}), z_w = 1, ..., r_w + p_w$, sont définis pour indiquer qu'aucun autre exemplaire du type w n'est chargé dans la série s du bloc non agrégé, sur le quai ou sur la rampe selon le cas, et les arcs $(I'^k_{w\,z_w\,s}, E^k_{ws}), z_w = 1, ..., r'_w$ ou $z_w = 1, ..., p'_w$ sont définis de la même façon, mais pour le bloc agrégé.

Un coût de 1 est associé à tous les arcs $(O^k, B^k_{w1}), w = 1, ..., |N^k|$ et un coût de 0 pour tous les autres arcs.

L'exemple suivant illustre la construction du réseau et la définition des contraintes de demande de chargement dans les équations (2.2).

Exemple 3. Supposons que nous devions charger les items décrits dans le tableau 3.7.

Tableau 3.7 – Données pour la construction du réseau

w	q_w	l_w	m_w	Description de l'item
1	4	267	170	Medium Logistics Vehicle Wheeled 2 $\frac{1}{2}$ ton
2	6	232	248	Armoured Personel Carrier M113.
3	10	109	11	$\frac{1}{4}$ Ton Trailer.

Comme la longueur du quai est de 492 pouces, nous pouvons y mettre un maximum de trois séries d'items, en alternant les items de type 2 et 3. Ainsi, le nombre de séries sur le quai est de trois. Seul les items de type 3 peuvent être mis sur la rampe, donc une série est accordée à ce type. Pour le premier type d'item, nous ne pouvons mettre qu'un item sur le quai. Le bloc non agrégé est constitué d'un seul item tandis que le bloc agrégé est constitué de trois items. Nous avons deux contraintes de demande associée à ce type d'item. La première de ces deux contraintes contient un seul item et le membre de droite sera 1. La deuxième contrainte englobe les trois autres items et le membre de droite sera égal à 3. Un nœud est associé respectivement à l'item du premier bloc et aux trois items du second. Pour le second type d'items, deux items vont former le bloc non agrégé car un maximum de deux items peuvent être mis l'un après l'autre pour ce type. Le bloc agrégé est constitué de quatre items. Nous associons deux contraintes avec membre de droite 1 pour le bloc non agrégé et une contrainte avec membre de droite 4 pour l'agrégé. Par ailleurs, deux nœuds vont représenter respectivement les deux items non agrégés et deux autres nœuds vont représenter les quatre items restants pour chaque série sur le quai. Pour le troisième type, quatre items peuvent être mis l'un après l'autre sur le quai et un sur la rampe. Chaque bloc respectif est constitué de cinq items. Cinq contraintes ayant 1 comme membre de droite et une contrainte ayant 5 comme membre de droite décrivent la demande pour le type. Pour chaque série sur le quai et sur la rampe, un nœud est associé à chacun des cinq premiers items. Pour le bloc agrégé, quatre autres nœuds

sont associés aux cinq items du bloc agrégé pour les séries sur le quai et un nœud est associé à ces cinq items pour l'unique série sur la rampe.

Pour adapter le branchement précédent à cette nouvelle formulation, nous prenons les décisions de branchement uniquement sur les variables de flot entre deux nœuds quelconques associés respectivement à un item et dont *au moins* l'un de ces items doit provenir d'un bloc non agrégé. Dans ce cas, nous sommes assurés que la valeur du flot entre ces deux nœuds est entre 0 et 1. Au début de l'algorithme, nous choisissons deux items pouvant se succéder, suivant un critère de sélection. Le premier est toujours le plus long de la liste et issu de son bloc non agrégé. Le second est un item candidat à charger à l'avant ou à l'arrière du premier. Nous procédons par la suite à des substitutions. D'abord, si le candidat est issu de son bloc agrégé ou non agrégé, nous le substituons avec un item de son bloc non agrégé et la règle de substitution développée dans la section précédente pour le cas non agrégé est appliquée. L'item de gauche aura le rang le plus grand parmi les items du bloc non agrégé encore disponibles, et l'item de droite aura le rang le plus petit parmi les items encore disponibles de son bloc non agrégé. Pour défaire la décision, il faut ajouter des contraintes d'interdiction de succession entre les items. Pour étendre une séquence d'items, nous adoptons la même stratégie que dans le cas non agrégé et nous utilisons ces règles jusqu'à l'obtention d'un plan de chargement. Lorsqu'un plan de chargement est trouvé, nous procédons à une dernière substitution. Les items du plan sont remplacés par des items issus du bloc agrégé du même type, de sorte qu'une mise à jour sur le membre de droite de la contrainte de demande est possible. Ainsi, les items du bloc agrégé sont les premiers à être réduits et un nouveau problème résiduel apparaît. La procédure est réappliquée au problème résiduel courant jusqu'à l'obtention de la solution optimale, car de telles décisions simplifient le problème sans altérer la structure du sous-problème.

Il est important de noter que, dans le modèle non agrégé de Desrochers et Soumis (1989), les décisions de fixation ou d'interdiction sont transmises du nœud père vers

tous ses descendants dans l'arbre de recherche, car le réseau ne contient qu'un seul bloc d'items et chaque item est spécifique. Par contre, pour notre modèle, une fois qu'un plan de chargement est trouvé, un nouveau problème résiduel est résolu et les décisions sur la fixation d'une suite composée d'un ou de plusieurs exemplaires d'items ne seront plus considérées. Cependant, si à un nœud quelconque de l'arbre de recherche, lors d'un retour en arrière, si l'interdiction de la *première* décision pour construire un nouveau plan est imposée, alors cette décision est conservée sur tous les nœuds descendants du nœud considéré. La décision est assimilée à la non considération d'un plan qui contient la séquence dans tous les réseaux résiduels issus des nœuds descendants dans l'arbre de recherche.

La même coupe que dans le second chapitre est utilisée pour renforcer la relaxation linéaire à travers l'arbre de recherche. Pour résoudre les instances dans les tests numériques présentés dans la section qui suit, l'arbre d'énumération est exploré en utilisant aussi la même procédure que dans le second chapitre.

3.7 Résultats numériques

L'algorithme de la section 3.6 a été implanté et testé sur le problème réel de Ng (1992) et ses dérivés, et les problèmes générés aléatoirement du chapitre précédent. Un problème additionnel tiré de celui de Ng où nous avons ajouté des priorités d'envoi est considéré. Les résultats sont commentés en utilisant les mêmes définitions qu'à la section 3.5.

Les résultats obtenus à partir des données de Ng et des instances dérivées sont présentés dans les tableaux 3.8, 3.9 et 3.10 qui décrivent respectivement les résultats pour différentes largeurs de l'intervalle d'admissibilité du CG.

Tableau 3.8 – Intervalle large pour le CG : (245 ;737)

Priorité	1 Priorité				4 Priorités			
modèle	Une	Deux	Trois	Quatre	Une	Deux	Trois	Quatre
CPUC	14	30	32	42	8	13	15	21
CPUA	202	1963	8715	22590	77	655	2963	8646
Sol	92	184	275	367	101	201	302	402
Relax	91.6	183.1	274.6	366.2	100.5	201.0	301.5	402.0
B&B	406	828	1239	1654	419	837	1002	1611
ColGenC	1509	2361	2363	2782	1157	1298	1203	1275
ColGenA	4293	9356	14150	19316	3026	6422	9515	13039
ArcsC	3256	3530	3554	3554	3940	4470	4518	4518
ArcsA	8159	14273	20387	26501	10476	19266	28056	36846
NœudsC	825	946	958	958	825	946	958	958
NœudsA	2291	4329	6367	8405	2291	4329	6367	8405
MemC	20.1	21.8	22.0	22.3	3.8	4.5	4.5	5
MemA	76.4	154.9	241.0	334.3	11.0	26.0	46.2	70.7

Tableau 3.9 – Intervalle moyen pour le CG : (551 ;564)

Priorité	1 Priorité				4 Priorités			
modèle	Une	Deux	Trois	Quatre	Une	Deux	Trois	Quatre
CPUC	45	34	45	92	8	14	21	20
CPUA	319	2410	9589	23680	83	805	3588	8335
Sol	92	184	276	368	106	212	318	424
Relax	92.0	184.0	276.0	368.0	106.0	212.0	318.0	424.0
B&B	229	118	587	903	340	679	1016	1353
ColGenC	2183	1805	1810	3243	970	1176	1457	975
ColGenA	9383	26127	56548	59151	3077	6771	9828	13899
ArcsC	3256	3530	3554	3554	3940	4470	4518	4518
ArcsA	8159	14273	20387	26501	10476	19266	28056	36846
NœudsC	825	946	958	958	825	946	958	958
NœudsA	2291	4329	6367	8405	2291	4329	6367	8405
MemC	16.1	17.4	17.6	17.9	3.5	4.4	4.5	4.7
MemA	57.8	119.7	189.2	263.2	10.7	25.9	45.9	76.2

Tableau 3.10 – Intervalle serré pour le CG : (559 ;563)

Priorité	1 Priorité				4 Priorités			
modèle	Une	Deux	Trois	Quatre	Une	Deux	Trois	Quatre
CPUC	40	63	96	95	9	17	25	25
CPUA	216	1503	5860	12770	93	869	3776	9747
Sol	97	194	291	388	108	216	324	432
Relax	96.8	193.7	290.6	387.4	108.0	216.0	324.0	432.0
B&B	400	839	1201	1675	343	684	1287	1362
ColGenC	2953	3662	4698	4913	1028	1523	1795	1592
ColGenA	7567	17863	33949	46319	3276	6735	10002	13665
ArcsC	3256	3530	3554	3554	3940	4470	4518	4518
ArcsA	8159	14273	20387	26501	10476	19266	28056	36846
NœudsC	825	946	958	958	825	946	958	958
NœudsA	2291	4329	6367	8405	2291	4329	6367	8405
MemC	13.5	14.9	15.1	15.5	3.5	4.2	4.5	4.8
MemA	46.7	97.9	157.7	222.9	10.7	25.2	44.0	68.4

Le premier constat est qu'il est possible de résoudre à l'optimalité le problème réel de Ng (colonnes marquées "Une") en moins d'une minute si aucune priorité d'envoi n'est fixée et en moins de 10 secondes avec quatre priorités sur l'ensemble des trois tableaux. De plus, le tiers des plans générés par le branchement standard est suffisant pour trouver la solution. Notons que dans cette première version de l'algorithme, les décisions qui consistent à fixer ou à interdire des inter-tâches de flot à une unité qui complètent un plan de chargement sont comptées dans le nombre total de nœuds de branchement parcourus pour résoudre chaque problème. Par ailleurs, un gain de mémoire consommée, d'au moins la moitié de celle de la méthode du chapitre précédent, est perçu.

La nouvelle formulation est moins sensible lorsque nous multiplions par deux, trois ou quatre fois les demandes de chaque item à transporter. Dans chaque tableau à l'exception de la première colonne du second, nous notons un accroissement linéaire du temps de résolution. Le temps maximal est inférieur à 2 minutes. Le nombre de nœuds de branchement et le nombre de plans générés semblent suivre une tendance

linéaire croissante. Par contre, le nombre d'arcs et le nombre de nœuds pour décrire chaque problème et la quantité de mémoire consommée tendent à rester constants lorsque le nombre total d'items croît de cette façon car leur valeur respective maximale sont atteintes lorsque la taille du problème est au moins le double de celle de Ng. En outre, comme le nombre de types d'items à transporter reste *constant*, le nombre de rangées dans le problème maître reste aussi constant, car c'est la demande qui croît. Ainsi, la difficulté de chaque problème, pour une priorité fixée, semble être la même. Pour cette raison, dans le processus de l'algorithme de la programmation dynamique, la mémoire utilisée et le temps CPU sont relativement les mêmes.

Pour les problèmes générés aléatoirement, les résultats sont présentés dans les tableaux 3.11 et 3.12. Ces problèmes sont les mêmes que ceux du chapitre précédent. Rappelons que le nombre de types d'items pour chaque problème est de 20 et que le nombre total d'items est toujours d'au moins 382. L'intervalle du CG est fixé à (551 ; 564) et aucune priorité n'est imposée. La moyenne et l'écart-type sont donnés dans les deux premières colonnes : MOY et $E.T.$

Tableau 3.11 – Problèmes aléatoires (partie 1)

	MOY	E.T	N1	N2	N3	N4	N5	N6
CPUC	420	852	61	37	53	1201	328	63
CPUA	1628	2015	340	224	763	4748	2197	552
Sol	145	26	135	120	151	103	151	140
Relax	144.2	25.7	134.8	119.4	151.0	102.6	150.8	139.3
B&B	628	102	499	504	641	570	735	576
ColGenC	4941	3548	2651	2752	2344	12395	6364	3060
ColGenA	9959	7873	4261	4237	6681	13511	9025	5445
ArcsC	3918	750	3206	3085	3267	4413	4135	3236
ArcsA	11059	2281	8249	7460	9331	10922	13048	9303
NœudsC	982	200	796	738	825	1168	1047	827
NœudsA	3272	698	2411	2143	2770	3212	3919	2764
MemC	42.1	38.1	18.3	4.6	38.4	96.7	44.9	12.5
MemA	270.7	346.4	59.2	18.9	191.8	913.8	291.7	55.2

Tableau 3.12 – Problèmes aléatoires (partie 2)

	N7	N8	N9	N10	N11	N12	N13	N14
CPUC	51	330	46	177	372	297	22	93
CPUA	510	1282	1145	704	1614	1949	227	1066
Sol	157	146	200	133	195	120	131	146
Relax	156.8	145.9	199.8	132.5	194.3	119.4	131.0	145.6
B&B	582	598	810	615	778	564	534	678
ColGenC	1710	2674	2234	4764	4406	4225	2050	4047
ColGenA	5475	5549	7092	6277	10722	9790	3976	6972
ArcsC	3752	5716	3242	4135	4767	3985	2985	3849
ArcsA	10686	14035	11407	10858	15527	10183	8089	12073
NœudsC	892	1419	819	1034	1171	980	730	937
NœudsA	3130	4058	3462	3184	4647	2959	2361	3594
MemC	34.4	91.0	19.6	39.3	38.6	51.7	14.1	17.0
MemA	165.7	358.1	95.6	223.1	243.6	417.0	48.8	89.2

	N15	N16	N17	N18	N19	N20	N21	N22
CPUC	324	164	209	885	736	34	48	479
CPUA	1144	823	1394	4370	1212	458	350	892
Sol	137	112	152	124	111	162	159	118
Relax	136.5	111.9	151.2	123.5	110.8	161.7	158.4	117.5
B&B	639	531	707	693	569	792	641	563
ColGenC	5323	5933	5395	10737	7968	2714	2769	7047
ColGenA	9523	19645	9934	30464	11951	4867	4329	10027
ArcsC	4941	3139	4100	3549	4262	2992	3875	3446
ArcsA	13094	8379	12667	10692	10726	9461	10848	8940
NœudsC	1229	769	1031	945	1093	743	955	899
NœudsA	3836	2451	3792	3227	3145	2815	3184	2643
MemC	45.7	16.6	29.4	71.3	68.6	6.9	19.2	46.8
MemA	191.9	113.4	167.0	751.9	273.6	32.1	57.5	196.7

	N23	N24	N25	N26	N27	N28	N29	N30
CPUC	8	4642	180	76	61	228	1329	63
CPA	175	9782	1021	338	472	1581	5781	1736
Sol	188	114	154	128	174	166	123	186
Relax	188.0	113.9	154.0	127.4	173.2	165.5	122.9	185.3
B&B	370	657	606	509	638	771	706	783
ColGenC	678	13734	2910	2130	2814	4952	14554	2916
ColGenA	3142	27489	8068	4177	5209	7089	35321	8516
ArcsC	2768	5300	4817	3836	3212	5063	4394	4094
ArcsA	8821	14471	13017	8966	9917	15556	13027	14048
NœudsC	647	1381	1197	916	811	1283	1145	1042
NœudsA	2600	4300	3812	2555	2967	4660	3912	4254
MemC	4.1	200.8	39.7	31.9	13.4	62.1	65.1	19.8
MemA	22.1	1708.1	145.5	81.1	38.9	332.7	740.3	97.4

Les résultats montrent que l'algorithme résout chaque problème dans un temps assez court par rapport à la méthode du chapitre précédent. L'algorithme résout en moyenne les problèmes quatre fois plus vite que la méthode non agrégée, même si le nombre de nœuds de branchement est toujours plus grand. Nous avons en moyenne des gains en termes de taille du réseau (3 fois moins) et de mémoire utilisée (5 fois moins). Le nombre de colonnes générées est à peu près la moitié de la méthode non agrégée sauf pour les problèmes les plus difficiles de la liste (N4, N18, N24, N29).

Pour terminer les tests, nous ajoutons un autre problème (tableau 3.13) dont la particularité est d'avoir un saut d'intègrité plus grand que l'unité. Les données du problème sont tirées de Ng. C'est un problème réduit à 126 items par rapport aux 322 originaux et les priorités d'envoi ont été construites à la main. Notons qu'il n'est pas facile de générer des problèmes qui répondent à ces critères car ces classes de problèmes sont rares et plus le nombre d'items augmente plus le saut diminue d'après les études de Bramel et Simchi-Levi (1993).

Tableau 3.13 – Les données tirées de celles de Ng

i	Qté	l_i (in.)	m_i ($\approx \times$100lbs)	p_i	Description de l'item
1	37	158	35	1	$\frac{1}{4}$ Ton Utility Truck
2	39	124	40	1	1 $\frac{1}{4}$ Ton Cargo Truck
3	3	232	248	4	Armoured Personel Carrier M113
4	2	109	11	3	$\frac{1}{4}$ Ton Trailer
5	17	166	54	1	1 $\frac{1}{2}$ Ton Trailer
6	1	58	30	2	Pallet 54" $\times$ 88"
7	27	224	94	1	1 $\frac{1}{4}$ Ton Command Post

Pour ce problème, les résultats de l'étude comparative sont donnés dans le tableau 3.14 dans lequel la solution optimale a été trouvée avec les deux méthodes de branchement. Le critère de Vanderbeck n'a pas pu prouver l'optimalité de la solution après avoir exploré un nombre maximal de 100 000 nœuds de branchement, bien que

Tableau 3.14 – Problème avec un *gap* plus grand que l'unité

Problème	Problème 1		
Méthode	Standard	Agrégé	Vanderbeck
CPU	4	1	600
Sol	36	36	36
Relax	35.1	35.1	34.7
B&B	162	157	100000
ColGen	1162	298	73720
Arcs	2185	616	348
Nœuds	751	224	119
Mem	1.7	0.5	4.2

celle-ci soit trouvée à moins de 1 seconde. Nous pensons que cela est dû d'une part à la borne inférieure liée à la formulation et d'autre part à la forte symétrie du problème. Le test montre que la formulation non agrégée (Standard) et la formulation mixte (Agregé) fournissent la meilleure borne par rapport à la formulation agrégée (Vanderbeck). Notons que le réseau utilisé dans chaque problème possède la même structure à l'exception des arcs et nœuds dans un groupe d'items quelconque selon que la formulation a pour second membre uniquement des 1, ou des nombres entiers plus grands que 1 ou les deux possibilités. Notons aussi que la mémoire consommée est plus grande avec la troisième méthode. Ces résultats montrent l'efficacité des méthodes de branchement exactes présentées pour résoudre la classe de problèmes et combattre les symétries sur les items. Nous ne donnons pas les résultats des tests utilisant la version non agrégée de la méthode proposée car les temps de calcul, les nombres de nœuds de branchement sont plus grands qu'avec la méthode standard à cause des procédures incorporées dans l'algorithme telles les permutations entre les exemplaires et les tests pour vérifier si un nœud doit être exploré ou non.

3.8 Extensions

Nous présentons une extension pour chaque algorithme présenté dans ce chapitre. Le critère de Vanderbeck (2000) peut être étendu en identifiant un ensemble de patrons qui utilise, au lieu d'un arc, un nœud bien précis du réseau et dont la somme des flots sur les arcs entrant est fractionnaire. Un nœud ne vérifiant pas cette condition est candidat au branchement et peut être retenu si le critère de choix est satisfait. Une borne inférieure ou supérieure, selon la branche considérée, est alors imposée à la somme des flots sur les arcs entrant au nœud candidat. Ces contraintes sont introduites au problème maître (2.14)-(2.16). Dans le sous-problème, des modifications des coûts sont effectuées à chaque arc entrant au nœud impliqué dans la prise de décision.

Dans le cas de la formulation non agrégée, une alternative de branchement serait de fixer dans une branche d'énumération un item à une position et d'interdire de le charger à cette position dans l'autre branche. Dans chaque branche de l'arbre d'énumération, ces décisions sont effectives en interdisant le passage de flot sur certains arcs, en l'occurrence ceux qui sont adjacents au nœud associé à l'item choisi selon les positions de ce dernier. De ce fait, la structure de problème de plus court chemin est toujours maintenue dans les deux branches. Le nombre de nœuds de branchement est fini car nous avons un nombre fini de positions et d'items. A chaque nœud de branchement, un item chargé en plusieurs positions différentes dans une solution optimale de la relaxation pourra être un candidat potentiel. Mais aucune garantie d'une solution entière n'est assurée par l'application de cette méthode même si tous les items sont à leur position optimale car les plans fractionnaires ne seront pas complètement éliminés. Par contre, combinée avec la méthode de branchement de Desrochers et Soumis (1989), elle devient exacte et peut même réduire le nombre de nœuds de l'arbre d'énumération. En effet, il suffit d'appliquer cette méthode de branchement à

chaque fois que nous commençons un nouveau plan. Une fois que le premier item du plan courant est fixé à une certaine position, le branchement de Desrochers et Soumis est utilisé pour chercher le prochain item à positionner ou à charger à l'avant ou à l'arrière de celui-ci jusqu'à l'obtention du plan complet. A ce point, l'alternative proposée s'interprète comme une version du branchement de Desrochers et Soumis, à la différence que la fixation d'une "inter-tâche" est effectuée en deux étapes : la fixation du premier item à une position suivie de la fixation des items qui lui sont adjacents. Elle est compétitive car elle dépend du nombre de nœuds associés aux items à transporter et des positions possibles de ces derniers au lieu du nombre d'inter-tâches dans le réseau qui fait la complexité du branchement de Desrochers et Soumis.

3.9 Conclusion

Nous avons présenté dans ce chapitre deux modèles agrégés obtenus après l'agrégation de certaines contraintes liant les items de mêmes caractéristiques. Chaque modèle admet une formulation utilisant des variables réelles et en nombres entiers. La décomposition de Dantzig-Wolfe est appliquée à chaque formulation pour obtenir une borne inférieure de la relaxation linéaire. Le problème maître comporte une seule contrainte de demande de couverture de chargement pour chaque item dans le premier modèle, rendant ainsi le second membre plus grand ou égal à 1. Dans le second modèle, le problème maître comporte, pour chaque item, un certain nombre de contraintes avec second membre fixé à 1, et éventuellement, une contrainte avec second membre plus grand ou égal à 1. La structure du sous-problème est la même dans les deux cas : celle d'un problème de plus court chemin avec contraintes de ressource. La méthode exacte de Vanderbeck (2000), qui devient une heuristique dans notre application, est utilisée pour résoudre le premier modèle. Tous les problèmes tests du chapitre précédent, tels le problème réel de Ng (1992) ainsi que les problèmes

dérivés construits à la main et les problèmes générés aléatoirement ont toutefois été résolus à l'optimalité. Les temps de calcul sont très courts, de l'ordre de 140 fois moins que l'agorithme présenté au premier chapitre, et utilisant peu d'espace mémoire, de l'ordre de 15 fois moins. Une nouvelle méthode de branchement est proposée. Elle permet de trouver la borne supérieure entière du second modèle de façon exacte. La règle s'inspire de celle de Desrochers et Soumis (1989). Elle élimine des problèmes de symétrie et améliore en même temps les décisions de branchement. Les résultats des tests montrent que l'algorithme résout aussi les instances présentées dans des temps satisfaisants avec un facteur de 4 fois moindre, et consommant 5 fois moins de mémoire par rapport à l'algorithme standard. Un dernier problème, de petite taille par rapport à tous les problèmes présentés, obtenu à partir du problème réel de Ng et comportant un *gap* plus grand que l'unité, est présenté pour tester le critère standard, celui de Vanderbeck et celui que nous avons proposé. Les résultats montrent que les deux approches trouvent rapidement la solution, alors que le critère de Vanderbeck n'arrive pas à prouver l'optimalité de la solution trouvée. A partir des tests effectués, nous déduisons la robustesse de notre algorithme. Les autres bénéfices du branchement incluent le renforcement des règles de branchement sans ajout de contraintes dans le problème maître. De plus, nous maintenons la structure d'un problème de plus court chemin avec contraintes de ressource dans le sous-problème, chaque fois qu'une décision de branchement est ajoutée. Le branchement est plus efficace que ceux qui fixent ou bornent une variable, et l'espace de recherche des solutions est mieux partitionné. Enfin, la complexité de l'algorithme ne croît pas à mesure que les décisions de branchement sont appliquées, ce qui permet l'accessibilité des solutions dans de grosses applications. Dans le prochain chapitre, nous allons voir comment les deux méthodes de branchement se comportent lorsque plusieurs villes sont visitées et lorsque les coûts de distance entre les villes sont pris en compte.

CHAPITRE 4 : MODÈLES AVEC PLUSIEURS VILLES DE COLLECTE

4.1 Introduction

Ce chapitre est consacré à l'étude du transport par avions-cargos de plusieurs items situés en des points géographiques différents vers un point particulier. Nous proposons un modèle à ce problème. Nous développons aussi un algorithme qui construit les plans de chargement réalisables pour tous les items, ainsi que les itinéraires suivis par les avions-cargos concernés, de telle sorte que les coûts totaux de transport soient minimaux. Les coûts totaux sont maintenant constitués des coûts variables dépendant des distances parcourues par les avions et des coûts fixes pour la sortie de chacun d'eux.

Le problème de chargement des avions-cargos se généralise en un problème de collecte et de livraison ou *PDAP (Pick-up and Delivery Aircraft's Problem)*. Chaque avion part d'une base et y rentre après avoir effectué une tournée; chaque item situé dans une ville bien précise doit être chargé exactement une fois dans un avion, en respectant certaines restrictions, entre autres, les priorités d'envoi, les contraintes opérationnelles de chaque avion, de même que les contraintes concernant le CG de l'ensemble avion/cargo. Si un avion doit ramasser des items dans les villes A, B et C, dans cet ordre, pour aller les déposer à une destination finale, les contraintes sur le CG doivent être respectées lors des vols suivants : entre les points A et B, entre les points B et C, et enfin entre C et la destination finale. Nous supposons que la distance à parcourir pour aller d'une ville A à une ville B est égale à celle pour aller de la ville B à la ville A, et que ces distances respectent l'inégalité triangulaire. Nous

supposons enfin qu'il faudra plus qu'un seul avion pour transporter tous les items, sinon le problème devient trivial. Nous supposons que chaque avion a un rayon d'action assez grand pour atteindre chaque ville et retourner à la base. Tout comme le problème de tournées de véhicule, ce problème est $\mathcal{NP}$-difficile et la recherche d'une solution optimale en un temps raisonnable peut s'avérer vaine.

Les développements effectués dans le réseau de base du chapitre précédent seront modifiés pour tenir compte des villes et des distances entre elles. Ainsi, nous avons agrégé les items, autrement dit, les items ayant les mêmes propriétés (longueur, poids, priorité, ville) sont considérés comme un seul type d'item dont le nombre de copies correspond à la demande pour ce type. Nous proposons les deux modèles suivants :

1. Dans un premier temps, nous supposons que les items sont chargés selon l'ordre de visite de leur ville de collecte. Ainsi, dans tout plan réalisable, les items de la première ville visitée sont chargés le plus près de la tête de l'avion, suivent les items de la deuxième ville et ainsi de suite, de sorte que la rampe est chargée, au besoin, par un item provenant de la dernière ville visitée. Nous parlerons alors d'un modèle sans déchargement.
2. Dans un second temps, nous permettons des réarrangements des positions entre les items déjà chargés dans les villes visitées. Nous parlerons alors d'un modèle avec déchargement. Cette procédure améliore ou facilite la faisabilité de la contrainte du CG de la cargaison.

La formulation unifiée que l'on retrouve pour résoudre les problèmes de tournées de véhicules et d'horaires d'équipages sert encore de modèle dans les deux cas et les contraintes de satisfaction des demandes sont représentées sous forme de partitionnement d'ensemble avec des variables entières. Le modèle est résolu par l'algorithme de séparation et évaluation progressive où les bornes inférieures sont calculées par génération de colonnes. Le problème maître comporte une ou plusieurs contraintes de couverture pour les items de même type rendant le second membre plus grand

ou égal à 1. La structure du sous-problème reste inchangée : celle d'un problème de plus court chemin avec contraintes de ressource. Pour le cas du problème sans déchargement, afin d'obtenir une "bonne solution" entière à chaque instance, nous pouvons utiliser soit le critère de Vanderbeck, soit les règles de branchement développées dans le chapitre précédent. Pour le cas du problème avec déchargement, les plans réalisables sont moins restrictifs mais des procédures d'interdiction de plans sont impliquées. Nous montrons les modifications effectuées pour que ces procédures soient efficaces. La méthode complète de résolution proposée est celle utilisant le critère de Vanderberck. Nous terminons ce chapitre par des tests numériques pour chaque cas présenté, prouvant que les approches que nous proposons donnent des résultats intéressants.

4.2 La revue de la littérature

La plupart des articles dont nous avons eu connaissance et traitant du problème de chargement des avions et/ou utilisant une formulation agrégée se restreignent au cas de transport d'une ville à une autre et/ou du problème du voyageur de commerce, autrement dit ils ne prennent en compte qu'un seul avion et/ou les coûts de distance n'interviennent pas. Nous les avons déjà présentés dans les chapitres précédents. Sur les articles traitant le problème de tournées de véhicule, Sierksma et Tijssen (1998) présentent une méthode heuristique basée sur la génération de colonnes pour résoudre les problèmes de rotation d'équipage sur des plates-formes pétrolières en Mer du Nord. Le transport des personnes à déplacer se fait par des hélicoptères et la demande de certaines plates-formes est réalisée par plusieurs hélicoptères. Ces quantités sont déterminées dans le sous-problème du processus de génération de colonnes. Une heuristique se basant sur une procédure d'arrondissement inférieur est utilisée pour obtenir une solution entière.

4.3 Un modèle sans déchargement aux escales

Le réseau $G^k = (V^k, A^k)$ correspondant à l'avion k décrit toujours les différentes possibilités de chargement de celui-ci. Les définitions et les interprétations des nœuds et des arcs sont les mêmes que dans les chapitres précédents. Cependant, comparé au modèle précédent, il y a une différence importante. Les items sont toujours regroupés par type, mais le type inclut maintenant la ville de provenance. Nous commençons par décrire les changements opérés dans le réseau de base pour tenir compte des villes et des distances entre elles. Nous donnons ensuite la formulation mathématique en précisant les modifications effectuées par rapport aux formulations des chapitres précédents.

4.3.1 La description du réseau

Les modifications principales sur le réseau sont dans la structure de coûts qui doivent tenir compte des déplacements de l'avion entre les villes. Ces coûts sont ajoutés sur les arcs $(E^k_{ws}, B^k_{w's+1}), w, w' = 1, ..., n, w \neq w'$, si les types w et w' sont dans des villes différentes. Si la série s est la dernière sur le quai et $s+1$ la première sur la rampe, on doit traverser le nœud *rampe*, et par conséquent on risque de perdre toutes traces du lien entre les deux villes. On associera donc un nœud *rampe* à chaque ville et le coût sera mis sur l'arc entre E^k_{ws} et le nœud *rampe* associé à la ville. Dans notre application, vu la grosseur des items à transporter, il n'y a qu'un seul item pouvant être mis sur la rampe. De plus, on voudrait éviter des chargements nécessitant plus d'une visite dans une même ville. Cela simplifiera notre réseau en imposant une seule série s sur la rampe, ce qui élimine tous les arcs interville. Les arcs (E^k_{ws}, D^k) ont aussi un coût. Le coût décrit le transport de tout le chargement à partir de la ville où l'item w a été chargé vers l'unique ville de livraison D^k et le retour de l'avion à sa base O^k.

4.3.2 La formulation de partitionnement d'ensemble

En utilisant les mêmes notations que dans la formulation originale, les arcs de coût non nul sont les arcs $(O^k, B^k_{w1}), w = 1, ..., |N^k|$ et les arcs qui permettent de passer d'une ville à une autre ou d'une ville à la rampe ou d'une ville au nœud D^k. Dans la suite, chaque type d'item $w \in N$ est différent pour chaque ville. La formulation de ALP s'écrit dans ce cas :

$$z_{IP} = \min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij} X^k_{ij} \tag{4.1}$$

sujet aux contraintes :

$$\sum_{k \in K} \sum_{s \in S} \sum_{z=1}^{r_w} \sum_{(i,j) \in A^k : i = I^k_{wzs}} a^k_{w,ij} X^k_{ij} = q_w, \forall w \in N \tag{4.2}$$

$$\sum_{j:(O^k,j) \in A^k} X^k_{O^k j} = \sum_{j:(j,D^k) \in A^k} X^k_{jD^k} = 1, \forall k \in K \tag{4.3}$$

$$\sum_{j:(i,j) \in A^k} X^k_{ij} - \sum_{j:(j,i) \in A^k} X^k_{ji} = 0, \forall k \in K, \forall i \in V^k \tag{4.4}$$

$$X^k_{ij} \in \{0,1\}, \forall k \in K, \forall (i,j) \in A^k \tag{4.5}$$

$$X^k_{ij}(f^{kr}_{ij}(\mathrm{T}^k_i) - T^{kr}_j) = 0, \forall k \in K, \forall r \in R, \forall (i,j) \in A^k \tag{4.6}$$

$$a^{kr}_i \leq T^{kr}_i \leq b^{kr}_i, \forall k \in K, \forall r \in R, \forall i \in \{O^k, D^k\} \tag{4.7}$$

$$a^{kr}_i \Big(\sum_{j:(i,j) \in A^k} X^k_{ij} \Big) \leq T^{kr}_i \leq b^{kr}_i \Big(\sum_{j:(i,j) \in A^k} X^k_{ij} \Big), \forall k \in K, \forall r \in R, \forall i \in V^k. \tag{4.8}$$

L'application de la décomposition de Dantzig-Wolfe à cette formulation multiflot nous permet d'étudier directement le problème à partir des plans de chargement réalisables. Il nous reste à préciser la fonction de prolongation qui nous permet de trouver les coûts de chaque plan.

Rappelons que nous avons cinq ressources pour évaluer à un noeud donné du réseau, respectivement la longueur totale de l'ensemble des items chargés, le poids total de cet ensemble, le poids des items sur la rampe s'il en existe, le CG de l'ensemble si on tasse les items vers l'avant de l'avion et le CG de l'ensemble si on tasse tous les items vers l'arrière. Dans la suite, nous ajoutons d'autres ressources pour tenir compte de la distance totale parcourue. Une des particularités de ce problème est le fait que le coût d'un plan de chargement dépend des villes visitées et des items chargés et ne retient qu'une seule fois le coût de passage dans chaque ville. Les nouvelles ressources ont pour tâches de mettre en valeur les villes visitées, et de déterminer si le prochain item à charger provient ou non de l'une de ces villes. Cela évite de revenir charger dans une ville après une tournée intermédiaire dans une ou plusieurs autres villes. Cette proposition est facile à modéliser dans la mesure où des variables binaires sont suffisantes pour gérer ces contraintes. Soit $\mathcal{V}$ l'ensemble des villes et $(T_i^1, T_i^2, .., T_i^{|\mathcal{V}|})$ le vecteur ressource ville qui est défini à chaque nœud i du réseau et qui a autant de composantes T_i^v, $v \in \mathcal{V}$ qu'il y a de villes. Pour chaque avion k, au nœud O^k, ce vecteur est nul. Chaque composante prend une valeur binaire 1 ou 0 pour exprimer le fait que la ville correspondante est visitée ou non. Pour chaque composante $T_i^v, v \in \mathcal{V}$ définie au nœud i, la fonction de prolongation correspondante $f_{ij}^{k(5+v)}$ s'écrit comme suit :

$$f_{ij}^{k(5+v)}(\mathbf{T}_i^k) = \begin{cases} T_i^{k(5+v)} + 1 & \text{si } j = B_{ws}^k \text{ et } w \text{ est situé dans } v, \\ T_i^{k(5+v)} & \text{sinon.} \end{cases} \tag{4.9}$$

De façon plus précise, cette consommation est de 1 unité, si on parcourt les arcs $(O^k, B_{w1}^k), w = 1, ..., |N^k|$, de début de chargement, les arcs $(E_{ws}^k, B_{w's+1}^k)$ $w, w' = 1, ..., |N^k|, w \neq w'$, liant deux types d'items entre deux villes distinctes, les arcs liant E_{ws}^k et les nœuds *rampe* ou les arcs (E_{ws}^k, D^k), $w = 1, ..., |N^k|$. Sinon, elle est 0.
Une borne inférieure de 0 et une borne supérieure de 1 permettent de limiter le

nombre de visites pour chaque ville v à chaque noeud j du réseau.

$$0 \leq T_j^{k(5+v)} \leq 1.$$

Dans un chemin partiel, la consommation de ressource cumulée, pour chaque composante du vecteur, se fait d'un noeud du réseau à un autre. Lorsque cette consommation dépasse sa borne supérieure, c'est que la ville de passage actuelle a déjà été visitée et la prolongation de ce chemin partiel n'est plus valide.

Le problème maître et le sous-problème ont déjà été définis dans l'introduction. La méthode de résolution de ces deux problèmes est la même que dans les chapitres précédents.

Dans notre modèle, la dernière fonction de prolongation, celle qui gère la borne inférieure de la contrainte du CG, n'est pas une fonction non décroissante. La proposition suivante permet de détecter les étiquettes dominées. La démonstration est analogue à celle de la proposition 3 du second chapitre.

Proposition 5 *Soient s et t deux chemins partiels définis à un nœud $i \in N$ avec respectivement comme vecteur ressource associé $\mathbf{T}_i^{k,s}$ et $\mathbf{T}_i^{k,t}$ et comme coût respectif c_i^s et c_i^t. L'état s domine l'état t si :*

$$\begin{aligned}
c_i^s &\leq c_i^t \\
T_i^{k1,s} &\leq T_i^{k1,t} \\
T_i^{k2,s} &= T_i^{k2,t} \\
T_i^{k3,s} &\leq T_i^{k3,t} \\
T_i^{k4,s} &\leq T_i^{k4,t} \\
T_i^{k5,s} &\geq T_i^{k5,t} \\
T_i^{k(5+v),s} &= T_i^{k(5+v),t}, v = 1, ..., |\mathcal{V}|.
\end{aligned}$$

4.3.3 La recherche d'une solution entière

Les méthodes de branchement sont celles présentées dans le chapitre précédent. Ces méthodes sont testées sur des instances qui présentent des symétries entre les items transportés et dont le saut de intègrité est assez grand. Pour la méthode de Vanderbeck, nous avons testé plusieurs critères pour sélectionner l'arc de branchement, entre autres, l'arc adjacent à l'item le plus long, l'arc dont la partie fractionnaire est le plus près de 0 ou le plus près de 0,5 ou le plus près de 1. Nous avons retenu celui qui donne les meilleurs résultats sur des instances cibles, à savoir l'arc dont la partie fractionnaire est le plus près de 0,5. Pour la méthode de branchement que nous avons développée, le critère de choix est le suivant. Pour la première inter-tâche à fixer lors de la création d'un nouveau chemin, celle qui est adjacente à l'item le plus long nous semble la plus satisfaisante. Les inter-tâches suivantes qui formeront le chemin sont celles dont le flot est le plus proche de la valeur 1.

L'efficacité de la recherche de la solution optimale dépend encore de la politique d'exploration de l'arbre de recherche. Dans notre application, il est préférable d'explorer en "profondeur d'abord" (*depth-first*) l'arbre pour trouver une solution entière rapidement. Ainsi, nous choisissons de chercher du côté de l'arbre où le flot passant sur un arc est mis à sa borne supérieure. Une fois qu'une solution entière est trouvée, on utilise comme politique d'exploration de l'arbre de recherche une recherche "meilleur d'abord" (*best-first*) c'est-à-dire en primant la meilleure borne inférieure. La recherche est terminée si la solution trouvée est réalisable et/ou un critère d'arrêt, que nous préciserons dans la section présentant les tests numériques, est satisfait.

Pour obtenir une solution réalisable ou une borne supérieure dans un temps relativement court, nous pouvons nous servir du modèle de base. L'idée serait de minimiser le nombre d'avions en mettant dans un premier temps tous les coûts de distance entre

les villes à 0 et une fois que la solution optimale est trouvée, recalculer les coûts de chaque plan optimal selon les villes visitées en utilisant les coûts réels. Dans toute la suite, nous appelerons cette heuristique l'algorithme *Coûts_Nuls*.

4.4 Un modèle avec possibilités de déchargements

Dans cette partie, nous proposons une alternative au modèle de chargement d'avions-cargos avec plusieurs villes de collecte et une seule ville de livraison. Dans ce modèle, nous supposons que des réarrangements des items déjà chargés dans les villes visitées sont possibles. Les principales difficultés auxquelles nous devons faire face se situent au niveau de la modélisation. Ainsi, supposons que nous ayons trois bases A, B et C où sont entreposés un nombre limité d'avions-cargos ainsi que certains items commandés et que ces trois sites satisfont à toutes les demandes. Soit D le site de livraison. Les contraintes du CG doivent être satisfaites entre chaque paire de bases visitées. De ce fait, le modèle construit doit tenir compte de la provenance de chaque item chargé, et il faut s'assurer qu'à chaque décollage, il existe une permutation des items chargés qui respecte les contraintes du CG.

Rappelons que dans le premier modèle, où aucun déchargement n'est possible, nous avons autant de ressources que de villes. Le modèle que nous présentons ne tient pas compte du nombre de villes, mais nous allons nous restreindre à quelques villes pour illustrer le modèle. Nous supposons, sans perte de généralité, que tous les avions se trouvent au même endroit, et que cette base constitue le lieu de livraison. A partir de la matrice des distances entre chaque paire de sites, on déduit les coûts de déplacement de la ville de livraison vers les autres sites, de même que le coût du retour à la base. Remarquons que le coût de chaque tournée n'est connu que lorsque nous connaissons les villes visitées et l'ordre de visite. Nous avons fixé chaque coût

à partir d'un calcul indépendant de l'optimisation. Ainsi, nous n'avons pas introduit de ressources pour modéliser ces coûts dans ce cas particulier.

Le réseau que nous testons est une extension de celui du chapitre précédent, à savoir celui de la section 3.2.1. Notons que le modèle peut s'adapter à tous les réseaux présentés dans ce travail. Dans la section suivante, nous détaillons les arcs, les coûts sur ces arcs et les nœuds de ce réseau. Des modifications sont apportées à l'intérieur de chaque groupe d'items car l'information sur la ville de provenance des items fait partie d'un critère de différence entre ces items. Les rôles des nœuds et des arcs vont être précisés ultérieurement.

4.4.1 La description du réseau

Comme dans les chapitres précédents, pour décrire les plans de chargement intéressants, nous avons besoin d'un réseau et à chaque réseau $k \in K$ nous associons un avion-cargo. Chaque item à transporter est en correspondance avec un nœud dans le réseau et ce nœud est copié autant de fois que le nombre de séries $s \in S$ le permet sur le quai et sur la rampe, pour décrire les différentes possibilités de placement de l'item dans l'avion. Les nœuds et les arcs du chapitre 3, paragraphe 3.2.1 sont retenus dans tout ce qui suit et leur fonction restent les mêmes. Seuls les arcs $(E^k_{ws}, D^k), w \in N, s \in S$ ont un coût non nul qui sera précisé plus loin. Notons qu'une demande q_w est associée au type d'item $w \in N$ et r_w nœuds représentent l'item. Une seule variable duale est en correspondance avec la contrainte. Il s'ensuit que les r_w représentants, indépendamment de leur placement $s \in S$ dans le réseau auront la même variable duale. Par conséquent, les arcs qui partent de ces nœuds auront aussi leurs coûts égaux.

4.4.2 La formulation mathématique

La formulation de *ALP* est la même que dans le cas sans possibilité de déchargement, c'est-à-dire qu'elle s'écrit avec la fonction objectif (4.1) et les contraintes (4.2)-(4.8). Les contraintes sur le CG auront un traitement particulier.

4.4.3 Une méthode heuristique de résolution

L'application de la décomposition de Dantzig-Wolfe à cette formulation multiflot nous permet d'étudier directement le problème à partir des plans de chargement réalisables. Nous avons déjà mentionné dans l'introduction les grandes lignes de la méthode. L'algorithme de génération de colonnes imbriqué dans une méthode de séparation et d'évaluation progressive est utilisé pour résoudre ce programme en nombres entiers.

Remarquons qu'une relaxation du problème original est obtenue en ne tenant pas compte des contraintes sur le CG entre les villes intermédiaires où sont ramassés des items. Les plans qui visitent une seule ville sont alors tous réalisables. Nous devons nous assurer de la faisabilité des plans qui visitent plus de deux villes de collecte. Pour y arriver, nous commençons par déterminer, pour chacun de ces plans, l'ordre de visite des villes le moins coûteux en terme de coût de déplacement. Nous vérifions, par la suite, si les contraintes sur le CG sont respectées. Les algorithmes d'Amiouny *et al.* (1992) ou de Mathur (1998) ou une énumération complète pourraient être utilisés dans ce cas. Nous avons opté pour une procédure plus simple. Pour retenir un plan, il suffit de résoudre l'ALP en tenant compte du CG des items chargés à chaque escale, moyennant des permutations de positions entre les items, et de voir si, un seul chargement est suffisant à chaque fois. Sinon, le plan non réalisable doit être interdit. Notons ici, qu'il est interdit mais qu'un ordre de visite différent permettrait peut-être

de rendre ce plan réalisable. Nous ne vérifierons pas cette éventualité et nous éliminerons ce plan qui est dorénavant qualifié de chemin interdit. Pour interdire un chemin lors de la génération de colonnes, nous tirons profit des travaux de Villeneuve et Desaulniers (2002). Le plus court chemin est maintenant défini dans un graphe entre un nœud *source* et un nœud *puits* sous la contrainte que ce chemin est différent de tout chemin contenu dans une liste ou ensemble de chemins interdits. Une procédure de pré-étiquetage, comme celle de Arunapuram, Mathur et Solow (1997), est associée à l'algorithme de Desrochers et Soumis (1989). Des pré-étiquettes sont apposées *a priori* à chaque nœud faisant partie de chaque chemin interdit, à l'exception du nœud *puits*. Chaque pré-étiquette représente un chemin partiel du chemin interdit allant du nœud *source* jusqu'au nœud associé à la pré-étiquette. Des informations additionnelles, telles les restrictions qui interdisent la prolongation du nœud faisant partie du chemin interdit, sont aussi contenues dans ces pré-étiquettes. Notons que, contrairement aux étiquettes, les pré-étiquettes possèdent des restrictions de prolongation, la dominance entre une étiquette et une pré-étiquette est autorisée, mais l'inverse ne l'est pas. En outre, une fois prolongées à un nœud valide, les pré-étiquettes deviennent des étiquettes standards. Enfin, la procédure pour détecter les étiquettes dominées est la même que lorsqu'il n'existe qu'une seule ville de collecte et une seule ville de livraison. Cette procédure a été présentée dans le deuxième chapitre.

Pour ce modèle, une solution entière est trouvée par le critère de Vanderbeck ou par l'heuristique *Coûts_Nuls*. Le critère de choix de l'arc de branchement et la politique d'exploration de l'arbre de recherche est la même qu'à la section 4.3.3. Une fois qu'une solution entière est trouvée, les tests de faisabilité des plans visitant plus d'une ville de collecte sont effectués. Au cas où un ou plusieurs plans non réalisables seraient détectés à partir de cette procédure, une réoptimisation avec un ensemble de plans interdits s'en suit. Une fois qu'une solution entière est trouvée, la politique d'exploration de l'arbre de recherche se transforme en une recherche "meilleur

d'abord" (*best-first*) c'est-à-dire en primant la meilleure borne inférieure. La recherche est terminée si la solution trouvée est réalisable et/ou un critère d'arrêt, que nous préciserons dans la section suivante, est satisfait.

4.5 Résultats numériques

Dans cette section, les modèles présentés précédemment sont testés sur des instances de problèmes de chargement des avions-cargos du Ministère de la Défense Nationale pour avoir une idée de sa flexibilité. Les deux modèles proposés sont comparés en termes de temps de résolution, de nombre d'avions requis pour chaque instance et de la qualité de la solution lorsque la méthode de génération de colonnes est appliquée. Pour le modèle sans possibilité de déchargement des items, les deux algorithmes présentés dans le chapitre précédent et qui tiennent compte des villes visitées sont testés. Pour le modèle avec possibilité de déchargement des items dans les villes intermédiaires, seul le critère de Vanderbeck est testé. A notre connaissance, il n'existe pas de jeux de données dans la littérature associés au problème de collecte et de livraison des avions-cargos tout en respectant les contraintes de centre de gravité. Par contre, nous avons déjà un jeu de données réelles et des instances générées aléatoirement et représentatives de problèmes réels pour une seule ville de collecte et une de livraison. Nous allons réutiliser ces données pour effectuer les tests numériques. Pour chaque instance, des informations additionnelles se rapportant sur la localisation de chaque item à transporter sont ajoutées. Les coûts sur les arcs sont calculés à partir des distances entres les villes concernées.

Si $l(i)$ et $L(i)$ (resp. $l(j), L(j)$) désignent respectivement les coordonnées de la latitude et de la longitude de la ville i (resp. de la ville j), alors le coût sur l'arc (i, j) se calcule comme suit :

$$c_{ij} = \left\lceil 10 * \arccos[\sin l(i) * \sin l(j) + \cos l(i) * \cos l(j) * \cos(L(i) - L(j))] * \frac{180}{\pi} \right\rceil.$$

Les coordonnées des villes sont données dans le tableau 4.1. Dans toute la suite, *Satolas* est choisie comme la ville de livraison et les autres sont des villes de collecte.

Tableau 4.1 – Les coordonnées des villes

Ville	Latitude	Longitude
Bagotville	48°19′50″	70°59′45″
Greenwood	44°59′4″	64°55′1″
Ottawa	45°19′21″	75°40′9″
Satolas	45°43′37″	5°4′55″
Trenton	44°7′8″	77°31′41″
Winnipeg	49°54′39″	97°14′36″

Comme les coûts de distance entre les villes sont supposés être des nombres entiers, les valeurs des c_{ij} sont multipliées par 10 et sont représentées par des entiers dans le programme informatique. Le coût de sortie d'un avion est de 100 000 unités et sa distance parcourue moyenne par tournée est le dixième de ce coût.

La sous-section 4.5.1 traite un cas particulier de problème pour lequel l'impact d'un coût élevé, associé à la visite de deux villes successives de collecte est étudié. La sous-section 4.5.2 étudie un problème de collecte entre trois villes de telle sorte que la situation géographique de la troisième ville varie. La sous-section 4.5.3 étudie le comportement des algorithmes proposés lorsque les items à transporter changent et lorsque le nombre de villes augmente. Les résultats numériques obtenus et les commentaires sont rapportés dans chaque sous-section respective. Finalement, les conclusions sont présentées dans la dernière sous-section.

4.5.1 L'impact d'un coût élevé entre deux villes de collecte

Cette sous-section analyse trois problèmes de collecte sur deux villes. Nous supposons qu'un nombre suffisant d'avions sont situés dans l'unique ville de livraison. Dans un

premier temps, chaque avion peut directement aller chercher les items à une ville de collecte et, éventuellement, à l'autre ville moyennant le coût réel de parcours entre les deux villes. Ensuite, l'étude se poursuit lorsque ce coût est fixé à un nombre suffisamment grand. Notons que les coûts associés à la visite d'une ville se traduisent sur chaque arc entrant au nœud correspondant à un item chargé dans cette ville.

Pour les tests effectués, le problème réel tiré de l'article de Ng (1992) est repris. L'intervalle ou la "fenêtre" du CG est fixé, pour une taille moyenne, à (551,564) lequel correspond à la contrainte réelle pour un *Hercules CC130* (Richardson, 1993). Aucune priorité d'envoi n'est imposée. Les données du problème de Ng contiennent un ensemble de 20 types d'items, et dans les tests effectués, les 10 premiers sont chargés, selon le cas, dans la ville de *Trenton* ou de *Winnipeg* ou de *Greenwood* et les 10 restants à *Bagotville* ou *Trenton* ou *Bagotville*, respectivement. Pour chaque modèle présenté et pour chaque problème étudié, la méthode utilisée est celle de Vanderbeck. Nous mentionnons le temps de résolution (CPU en secondes), la distance totale parcourue (Distance), le nombre d'avions (NbAvions), le critère d'arrêt (Arrêt) et le saut d'intégralité (Gap) de la solution obtenue exprimé en pourcentage.

Un critère d'arrêt, valable pour tous les tests numériques effectués, permet d'arrêter la résolution de chaque problème. Soit la solution trouvée est à l'intérieur d'un certain pourcentage de la valeur du *LP* au nœud racine de l'arbre de recherche et ce pourcentage est fixé à 0.15% pour les séries de tests, soit tous les nœuds de l'arbre de branchement ont été explorés. Dans ces cas, nous les notons par OPT et nous disons que la solution est "optimale". Soit une limite de temps de 5 400 secondes est dépassée, soit une limite de 1 000 nœuds de branchement parcourus est dépassée après qu'une première solution entière soit trouvée. Dans ces cas, nous les notons par LIM.

Pour chaque modèle, nous notons par *Réel* le problème dont les coûts de distance sont les distances réels entre les villes. Lorsque ces coûts de distance sont nuls, le problème

est noté *Nul* et l'heuristique *Coûts_Nuls* est utilisée. La solution obtenue est une approximation du problème original. En effet, une fois que sa solution optimale est trouvée, les coûts de chaque plan sont remplacés par les coûts réels dans la solution. Le saut d'intégralité est alors le quotient de la différence entre la solution obtenue lorsque les coûts réels sont pris en compte et la relaxation du problème original et de la relaxation du problème original. Par la suite, lorsque le coût de parcours entre les deux villes de collecte est fixé à un nombre suffisamment grand, le problème est noté *Grand*. Enfin, nous proposons à la colonne notée *Priorité* une alternative pour résoudre le problème noté *Grand*. Elle consiste à ajouter des priorités sur les items de l'instance et à faire appel à l'heuristique *Coûts_Nuls* pour sa résolution.

Tableau 4.2 – Étude sur les deux villes de collecte : *Trenton-Bagotville*

Modèle	**Modèle sans déchargement**				**Modèle avec déchargement**			
Coût	Réel	Nul	Grand	Priorité	Réel	Nul	Grand	Priorité
CPU	446	4	325	3	2	3	252	1
Distance	102508	103130	114758	114758	102020	102386	114758	114758
NbAvions	93	93	107	107	92	92	107	107
Arrêt	LIM	OPT	LIM	OPT	OPT	OPT	LIM	OPT
Gap	0.49	0.50	0.27	0.27	0	0.03	0.27	0.27

Pour le cas des deux villes *Trenton-Bagotville* (voir tableau 4.2) et pour le premier modèle, l'algorithme n'a pas pas réussi à montrer l'optimalité de la solution trouvée pour le problème *Réel* et le problème *Grand*. Cela est dû probablement à la piètre borne inférieure. Par contre, la résolution du problème *Nul* par l'heuristique *Coûts_Nuls* est très rapide et elle donne une approximation avec un saut d'intégralité 0,5% de la solution réelle. De même, le problème *Priorité* a été résolu rapidement. Empêcher les avions de parcourir deux villes l'une à la suite de l'autre fait passer le nombre d'avions de 93 à 107. Ce résultat est assez évident, mais la difficulté reste toujours dans la résolution du problème lorsque les coûts ne sont pas tous nuls. En faisant appel aux priorités d'envoi sur les items à transporter, nous avons pu contourner cette difficulté. Ainsi, les items de la première ville de collecte ont une priorité de

1 et les items de la deuxième ville de 3, de sorte qu'aucun plan contienne simultanément des items des deux villes. Les coûts de distance sont tous mis à 0 et on minimise le nombre d'avions. Nous utilisons alors l'heuristique *Coûts_Nuls* pour en déduire la solution. Les temps de calcul sont mieux dans le second modèle que le premier. De plus, le second modèle permet d'économiser un avion pour le problème *Réel* et la solution optimale a été trouvé pour trois problèmes. Pour le problème *Grand*, nous retrouvons la même difficulté de résolution que dans le premier modèle.

Tableau 4.3 – Étude sur les deux villes de collecte : *Winnipeg-Trenton*

Modèle	**Modèle sans déchargement**				**Modèle avec déchargement**			
Coût	Réel	Nul	Grand	Priorité	Réel	Nul	Grand	Priorité
CPU	4	4	333	3	2	3	317	1
Distance	119702	121234	129558	129558	120116	120776	129558	129558
NbAvions	93	93	107	107	92	92	107	107
Arrêt	OPT	OPT	LIM	OPT	OPT	OPT	LIM	OPT
Gap	0.48	0.49	0.27	0.27	0	0.01	0.27	0.27

Tableau 4.4 – Étude sur les deux villes de collecte : *Greenwood-Bagotville*

Modèle	**Modèle sans déchargement**				**Modèle avec déchargement**			
Coût	Réel	Nul	Grand	Priorité	Réel	Nul	Grand	Priorité
CPU	6	4	367	3	3	3	231	1
Distance	92752	93370	103278	103278	92570	92663	103278	103278
NbAvions	93	93	107	107	92	92	107	107
Arrêt	OPT	OPT	LIM	OPT	OPT	OPT	LIM	OPT
Gap	0.49	0.49	0.27	0.27	0	0.00	0.27	0.27

Dans les deux autres séries de tests (tableaux 4.3 et 4.4), nous notons que chaque problème réel est résolu à l'optimalité. De plus, l'heuristique *Coûts_Nuls* trouve une solution avec un saut d'intégralité relativement proche de la solution réelle. A partir de ces tests, nous pensons que les modèles présentés sont assez flexibles pour s'adapter à ces cas particuliers et générer une bonne solution.

4.5.2 L'impact de la variation de la géographie des villes

Cette partie analyse le comportement de l'algorithme lors d'une variation de la géographie des villes et/ou des coûts associés aux aéroports. Un problème de collecte entre trois villes est étudié de telle sorte qu'une des trois villes, toujours la même, change d'une instance à une autre ou son coût de passage augmente.

Pour les tests effectués, le problème réel tiré de l'article de Ng (1992) est repris. L'original des données a été modifié et complété comme dans la section précédente. Les données contiennent un ensemble de 20 types d'items, et dans les tests effectués, les 6 premiers sont chargés dans la ville *Trenton*, les 6 suivants à *Ottawa* et le reste à *Greenwood*. La ville *Ottawa* est remplacée par la ville *Bagotville* dans un premier temps et par la ville *Winnipeg* dans un second temps. Par la suite, la ville *Ottawa* est gardée mais un grand coût variant de 1 000 à 20 000 est ajoutée lors du passage à cette ville. Pour chaque modèle présenté et pour chaque problème étudié, la méthode utilisée est celle de Vanderbeck.

Tableau 4.5 – Étude de la géographie des villes et/ou des coûts d'aéroports

Modèle	**Modèle sans déchargement**							
Ville	*Ottawa*	*Bagotville*	*Winnipeg*	+1000	+2000	+5000	+10000	+20000
CPU	19	61	5	4	5	3	4	4
Distance	59543	60761	68306	101767	142767	265767	361109	560989
NbAvions	94	94	94	94	94	94	95	96
Arrêt	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT
Gap	0.35	0.36	0.35	0.32	0.27	0.13	0.06	0.00
Modèle	**Modèle avec déchargement**							
Ville	*Ottawa*	*Bagotville*	*Winnipeg*	+1000	+2000	+5000	+10000	+20000
CPU	3	3	3	3	3	3	391	375
Distance	51742	51860	60284	89742	127742	241742	372257	692257
NbAvions	92	92	92	92	92	92	93	93
Arrêt	OPT	OPT	OPT	OPT	OPT	OPT	LIM	LIM
Gap	0	0	0	0	0	0	0.49	0.34

Le tableau 4.5 montre que, dans l'ensemble des deux modèles, l'algorithme a réussi à trouver la solution optimale sur 14 problèmes en moins de 62 secondes environ.

Dans les deux derniers problèmes, le saut d'intégralité est relativement petit même si la recherche d'une solution réalisable devient plus difficile. Pour conclure cette série de résultats, nous montrons sur les courbes de la figure 4.1 l'évolution du nombre d'avions requis en fonction des coûts pour les deux modèles. Ces tests prouvent donc que si les coûts associés aux distances ou aux aéroports ne sont pas trop grands, il est possible de les résoudre de manière optimale dans des temps très raisonnables et qu'une borne supérieure sur le nombre d'avions est calculable dans les cas extrêmes.

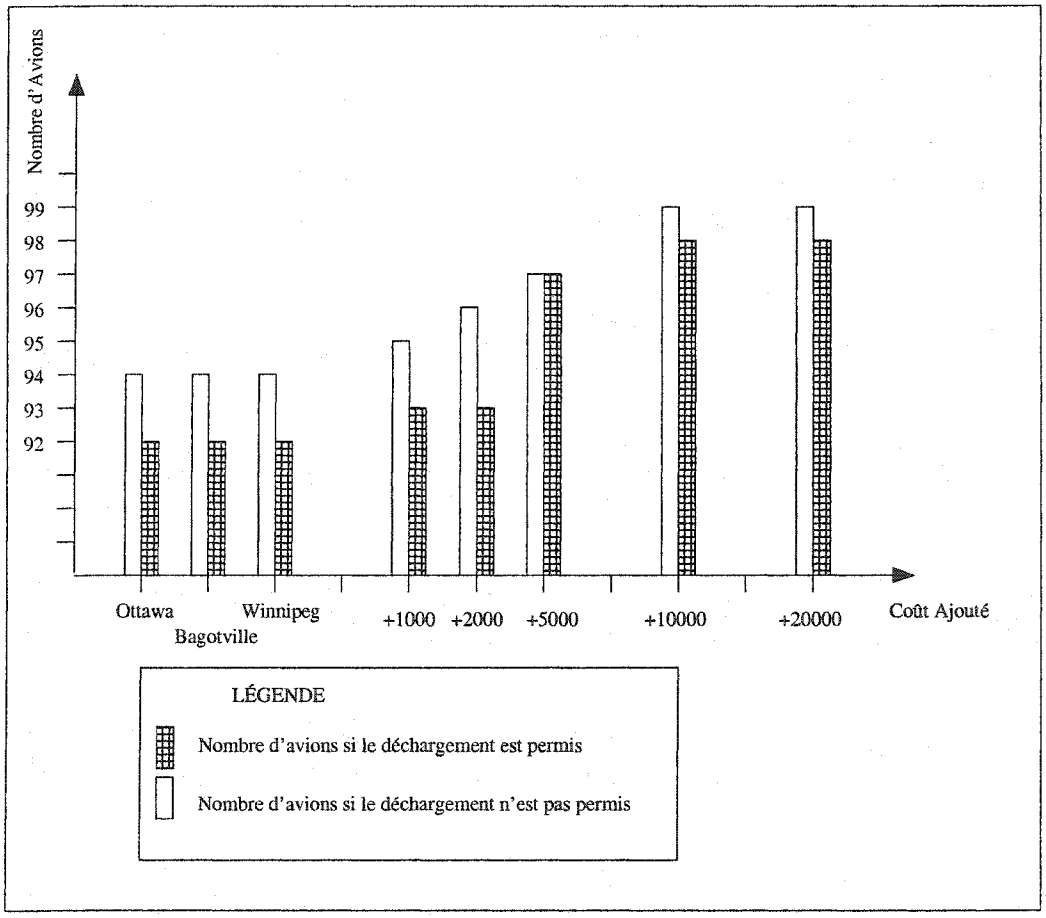

Figure 4.1 – Évolution du nombre d'avions lorsque des coûts sont ajoutés à la visite de la même ville

4.5.3 L'impact de la variation des items et du nombre de villes

Dans cette sous-section, une série de tests étudie le comportement des algorithmes développés dans le chapitre précédent sur le modèle avec possibilités de changements de position sur les items chargés et sur le modèle sans possibilité. Pour chaque modèle et pour chaque instance, les items à transporter changent. De plus, le nombre de villes de collecte varie entre trois et cinq. Les données, dans ces cas-là sont les mêmes que celles que nous avons utilisées dans les chapitres 2 et 3, à savoir les instances générées aléatoirement et représentatives de problèmes réels. Les 10 premières instances sont utilisées pour étudier le cas de trois villes de collecte : *Trenton, Ottawa, Greenwood*; les 10 suivantes pour le cas de quatre villes : *Trenton, Ottawa, Greenwood, Bagotville*; et les 10 restantes pour le cas de cinq villes : *Trenton, Ottawa, Greenwood, Bagotville, Winnipeg.* Dans tous les cas, la ville *Satolas* est choisie comme ville de livraison. Pour chaque instance, les données sont complétées par les villes de collecte pour les 20 types d'items. Dans le cas des problèmes comportant trois villes, les 6 premiers items sont chargés dans la première ville, les 6 suivants dans la seconde et les items restants dans la dernière ville. Pour les problèmes comportant quatre villes, les 5 premiers items sont chargés dans la première ville, les 5 suivants dans la seconde et ainsi de suite. Pour les problèmes à cinq villes, les 4 premiers items sont chargés dans la première ville, les 4 suivants dans la seconde et ainsi de suite. Aucune priorité d'envoi n'est imposée pour tous les tests. Pour le modèle sans déchargement, les trois méthodes développées dans le chapitre précédent sont testés pour chaque instance, à savoir, la méthode de Vanderbeck, l'heuristique *Coûts_Nuls* et la méthode mixte. Pour le modèle avec déchargement, la méthode de Vanderbeck et l'heuristique *Coûts_Nuls* sont testés pour chaque instance.

Le tableau 4.6 montre les résultats des tests obtenus à partir du modèle sans possibilité de déchargement. Le critère de Vanderbeck est testé, dans un premier temps,

Tableau 4.6 – Problèmes aléatoires (partie 1)

Modèle	Modèle sans déchargement									
Instance	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**I8**	**I9**	**I10**
CPU	31	5	27	3231	1549	365	1179	4053	6	101
Distance	77968	69234	91080	63722	87769	86518	91633	87665	118131	75568
NbAvions	136	122	153	106	154	141	160	148	200	134
Arrêt	OPT	OPT	OPT	LIM	LIM	LIM	LIM	LIM	OPT	OPT
Gap	0.55	0.75	0.54	0.77	0.45	0.23	0.24	0.58	0.08	0.00
CPU	5	4	10	42	35	2	9	35	5	9
Distance	80121	74016	97021	68350	89550	88838	97745	92429	120864	85723
NbAvions	136	122	153	106	154	141	160	148	200	134
Arrêt	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT
Gap	0.56	0.79	0.58	0.82	0.46	0.24	0.27	0.60	0.09	0.08
CPU	751	426	2219	5408	3321	661	518	4444	19	140
Distance	77968	69234	91080	64230	87773	86518	91585	87658	119030	75297
NbAvions	136	122	153	106	154	141	160	148	200	134
Arrêt	LIM	LIM	LIM	LIM	LIM	LIM	LIM	LIM	OPT	OPT
Gap	0.55	0.75	0.54	0.78	0.45	0.23	0.24	0.57	0.08	0.00
Instance	**I11**	**I12**	**I13**	**I14**	**I15**	**I16**	**I17**	**I18**	**I19**	**I20**
CPU	12	14	4	522	101	8	897	88	32	232
Distance	127007	76828	82168	97252	85737	71851	98708	78641	70645	95095
NbAvions	198	121	131	153	138	117	159	131	114	167
Arrêt	OPT	OPT	OPT	LIM	OPT	OPT	LIM	OPT	OPT	LIM
Gap	0.32	0.08	0.00	0.48	0.39	0.10	0.39	0.28	0.10	0.45
CPU	9	22	2	4	10	7	7	27	58	2
Distance	132561	79813	88653	99662	92051	73013	98900	82101	72298	102520
NbAvions	198	121	131	153	138	117	159	131	114	167
Arrêt	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT
Gap	0.34	0.10	0.05	0.50	0.44	0.11	0.39	0.30	0.11	0.50
CPU	42	34	25	760	1488	45	1726	5308	126	272
Distance	127007	77155	81799	97252	85737	69768	99068	78676	70645	94868
NbAvions	198	121	131	153	138	117	159	131	114	167
Arrêt	OPT	OPT	OPT	LIM	LIM	OPT	LIM	LIM	OPT	LIM
Gap	0.32	0.08	0.00	0.48	0.39	0.08	0.39	0.28	0.10	0.45
Instance	**I21**	**I22**	**I23**	**I24**	**I25**	**I26**	**I27**	**I28**	**I29**	**I30**
CPU	8	2142	2	5408	1468	10	6	1968	2912	95
Distance	120845	81656	121424	85661	112140	80343	125287	104707	86974	145157
NbAvions	159	120	191	119	155	131	175	167	124	192
Arrêt	OPT	LIM	OPT	LIM	LIM	OPT	OPT	LIM	LIM	OPT
Gap	0.35	0.43	0.26	0.74	0.34	0.54	0.29	0.34	0.31	0.47
CPU	10	12	1	592	9	7	7	18	28	21
Distance	125190	90210	128268	91832	119290	87894	134012	113244	93281	153021
NbAvions	159	120	191	119	155	131	175	167	124	192
Arrêt	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT
Gap	0.38	0.50	0.29	0.79	0.38	0.59	0.34	0.40	0.36	0.51
CPU	14	2637	4	5414	1550	24	13	2286	5026	1521
Distance	120845	81825	121424	84615	112105	80343	125287	104793	87410	148942
NbAvions	159	120	191	119	155	131	175	167	124	192
Arrêt	OPT	LIM	OPT	LIM	LIM	OPT	OPT	LIM	LIM	LIM
Gap	0.35	0.43	0.26	0.73	0.34	0.54	0.29	0.35	0.31	0.49

sur le modèle et à chaque fois une bonne solution est trouvée. En effet, sur toutes les instances, le saut d'intégralité maximum est de 0,8%. Pour le problème *I24*, probablement un des plus difficiles de la liste, une solution a été trouvée avec le crière d'arrêt imposé. L'heuristique *Coûts_Nuls* est utilisée sur ces mêmes instances. Cela constitue la deuxième série de tests et l'heuristique trouve en un temps relativement rapide à chaque fois le nombre d'avions optimal. Lorsque les coûts réels sont ajustés à la solution trouvée, le tableau rapporte aussi des sauts d'intégralité de moins de 0,8%. La méthode mixte est enfin testée sur les mêmes instances. Une solution a été trouvée pour chaque instance et le saut maximum est de 0,8%. Nous remarquons que la méthode prend plus de temps que celle de Vanderbeck, mais dans certains cas, elle trouve une meilleure solution. Ce qui la rend compétitive.

Les résultats des tests effectués sur le modèle avec possibilité de changements de position sont présentés dans le tableau 4.7. Dans les premières séries de tests, la méthode de Vanderbeck trouve toujours une solution et résout chaque instance en moins d'une heure avec un saut de 1,7% maximum. L'heuristique *Coûts_Nuls* est reprise avec les mêmes instances et, à chaque fois et dans un temps relativement rapide, le nombre optimal d'avions est trouvé. Lorsque les coûts réels sont ajustés à la solution trouvée, le tableau rapporte aussi des sauts d'optimalité de moins de 1,7%. Dans les instances *I4*, *I8* et *I9*, cette méthode trouve une meilleure solution que la première. Cela est dû probablement au fait qu'une solution réalisable est trouvée rapidement avec la première méthode et, lorsque la procédure de recherche primant la meilleure borne (*best-first*) est appliquée par la suite, le critère d'arrêt imposé vient à bout de la recherche.

4.6 Conclusions

Le problème du transport par avion-cargos de plusieurs items situés en des points géographiques différents vers un point particulier a été abordé dans ce chapitre. Les

Tableau 4.7 – Problèmes aléatoires (partie 2)

Modèle	Modèle avec déchargement									
Instance	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**I8**	**I9**	**I10**
CPU	12	134	8	3346	19	129	52	2588	130	911
Distance	75746	67593	85141	58753	84997	78881	88338	83421	112850	74784
NbAvions	135	120	151	104	151	140	157	148	201	133
Arrêt	OPT	LIM	OPT	LIM	OPT	LIM	OPT	LIM	LIM	LIM
Gap	0.19	0.51	0.00	1.38	0.11	0.50	0.08	1.43	0.58	0.38
CPU	3	1	7	35	15	2	4	28	3	8
Distance	76149	67983	85544	58319	85258	79117	88597	83082	112973	75229
NbAvions	135	120	151	103	151	140	157	147	200	133
Arrêt	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT
Gap	0.19	0.51	0.00	0.41	0.12	0.50	0.08	0.74	0.09	0.38
Instance	**I11**	**I12**	**I13**	**I14**	**I15**	**I16**	**I17**	**I18**	**I19**	**I20**
CPU	21	1056	3	344	576	17	658	2607	850	45
Distance	112432	67516	73721	83950	77605	64837	86430	70386	63436	90886
NbAvions	195	120	131	148	138	113	152	125	111	162
Arrêt	OPT	LIM	OPT	LIM	LIM	OPT	LIM	LIM	LIM	LIM
Gap	0.37	0.49	0.00	1.62	1.06	0.93	0.54	1.23	0.20	0.16
CPU	8	12	2	5	13	8	7	50	31	3
Distance	113051	68109	75009	82787	79373	65538	87432	71065	64305	92074
NbAvions	195	120	131	148	138	113	152	125	111	162
Arrêt	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT
Gap	0.38	0.50	0.01	1.61	1.07	0.93	0.55	1.23	0.21	0.16
Instance	**I21**	**I22**	**I23**	**I24**	**I25**	**I26**	**I27**	**I28**	**I29**	**I30**
CPU	6	1441	2	238	6	1217	18	507	80	9
Distance	108800	79133	114282	78984	105430	77422	119952	101519	81515	124339
NbAvions	159	118	188	114	154	128	174	166	123	186
Arrêt	OPT	LIM	OPT	OPT	OPT	LIM	OPT	LIM	OPT	OPT
Gap	0.36	0.45	0.00	0.05	0.02	0.44	0.46	0.31	0.08	035
CPU	4	9	1	139	7	6	3	10	28	6
Distance	116702	85049	121679	81150	107517	79295	129075	110316	93909	137943
NbAvions	159	118	188	114	154	128	174	166	123	186
Arrêt	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT
Gap	0.41	0.50	0.04	0.06	0.03	0.46	0.51	0.36	0.18	0.42

coûts de distance entre les villes et les contraintes opérationnelles de chargement sont pris en compte. Nous avons proposé deux modèles qui construisent des plans de chargement réalisables pour tous les items, ainsi que les itinéraires suivis par les avion-cargos concernés, de telle sorte que les coûts totaux de transport soient minimisés. Dans le premier modèle, les items sont chargés selon l'ordre de visite de leur ville de collecte et aucun arrangement de position n'est possible entre les items chargés dans les villes intermédiaires. Un second modèle qui permet des réarrangements de position entre les items déjà chargés dans les villes visitées est ensuite présenté. Ce dernier est moins restrictif que le modèle précédent et permet de considérer plus de plans réalisables, mais implique des procédures d'interdiction de plans. Une formulation sous forme de partitionnement ainsi qu'une approche, basée sur la méthode de génération de colonnes sont proposées pour chaque modèle. Pour trouver une "bonne solution" entière à chaque problème, nous utilisons soit la méthode de Vanderbeck, soit la méthode développée dans le chapitre précédent et adaptée à une formulation mixte. Le jeu de données réelles et les instances générées aléatoirement et représentatives de problèmes réels sont réutilisés pour effectuer les tests numériques.

Un cas particulier de problème pour lequel l'impact du coût élevé associé à la visite de deux villes successives de collecte est présenté et étudié. Chaque modèle peut supporter ces cas particuliers et l'algorithme utilisé trouve une bonne solution rapidement, mais une difficulté persiste, dans certains cas, pour prouver l'optimalité de la solution obtenue. Lorsqu'il s'avère très coûteux de visiter successivement les deux villes de collecte, le modèle avec priorité d'envoi des items permet de trouver rapidement la solution optimale.

Une autre étude se concentre sur l'impact de la géographie des villes de collecte. Dans l'ensemble des deux modèles, la méthode de Vanderbeck a réussi à trouver la solution optimale sur les 14 problèmes en moins de 62 secondes environ. Dans les deux derniers problèmes, le saut d'intégrité est relativement petit même si la recherche d'une solution réalisable devient plus difficile.

Enfin, une étude du comportement des algorithmes proposés, lorsque les items à transporter varient et lorsque le nombre de villes augmente, termine les séries de tests. Pour le modèle sans possibilité de déchargement, la méthode de Vanderbeck trouve pour chaque instance une bonne solution avec un saut d'intégralité maximum de l'ordre de 0,8%. L'heuristique *Coûts_Nuls* trouve à chaque fois en un temps relativement rapide le nombre d'avions optimal et, lorsque les coûts réels sont ajustés à la solution trouvée, les sauts d'intégralité sont au maximum de l'ordre de 0,8%. Avec la méthode mixte, nous obtenons aussi des solutions de même qualité, sinon meilleure, mais avec plus de temps de résolution, ce qui la rend compétitive. Les résultats des tests effectués sur le modèle avec possibilité de changements de position sont aussi satisfaisants. Dans les premières séries de tests, la méthode de Vanderbeck trouve toujours une solution et résout chaque instance en moins d'une heure avec un saut de 1,7% maximum. L'heuristique *Coûts_Nuls* trouve à chaque fois le nombre optimal d'avions dans un temps relativement rapide. Lorsque les coûts réels sont ajustés à la solution trouvée, les sauts sont de l'ordre de moins de 1,7%. Dans certaines instances, cette méthode s'avère plus performante.

Les résultats obtenus démontrent que les modèles proposés se résolvent assez bien avec la méthode de génération de colonnes. Pour chaque instance, une bonne solution a été trouvée. De plus, pour la classe de problèmes étudiés, l'heuristique *Coûts_Nuls* nous a permis d'obtenir une bonne solution dans un temps relativement court. Dans la recherche de la solution optimale, cette procédure est utile pour couper certaines branches non prometteuses d'un arbre de recherche. De plus, comme l'on s'y attendait, le second modèle permet d'économiser un ou plusieurs avions par rapport au premier et de réduire le temps de résolution de chaque problème.

CONCLUSION

Nous nous sommes intéressés au problème de chargement des avions-cargos. Plus particulièrement, nos travaux portent sur le développement d'un système de planification et d'optimisation pour cette classe de problème. L'originalité de cette thèse n'est pas la résolution d'un problème complexe mais la tentative d'une approche de résolution par génération de colonnes, technique qui a fait ses preuves dans le domaine des tournées de véhicules. Nous avons étudié le cas du problème entre une ville de collecte et une ville de livraison et nous nous sommes attaqués au cas de plusieurs villes de collecte par la suite.

Notre contribution porte principalement sur la modélisation et la résolution de ces problèmes de chargement. Une des clés pour exploiter au mieux l'approche est d'avoir un réseau approprié, de taille convenable en termes d'arcs et de nœuds, qui génère les "bons" chargements. Cela a été rendu possible, pour le cas étudié, par l'étude des connexions entre deux nœuds quelconques du réseau. Nous avons montré aussi qu'il était possible de tenir compte dans le modèle des contraintes opérationnelles de chargement et du balancement de l'ensemble avion et cargo. Pour cela, nous avons proposé des variables de ressource qui modélisent les contraintes locales de chargement en les rendant plus réalistes. Mais ce qui est intéressant également est que nous avons proposé un critère de dominance pour réduire les étiquettes dominées de telle sorte que l'algorithme de résolution du problème ne perde pas des solutions potentielles. L'arsenal de techniques développées par les chercheurs qui se sont penchés sur les problèmes classiques de tournées de véhicules est utilisé par la suite. Le modèle est résolu par la méthode de génération de colonnes imbriquée dans un algorithme de *B&B* en utilisant la méthode de branchement de Desrochers et Soumis (1990). Des tests ont été effectués pour étudier la performance de l'approche. Les résultats

montrent que le problème réel de Ng (1992) est résolu dans un temps raisonnable (en moins de 6 minutes). Une réduction de $2 millions est réalisé par rapport au coût trouvé par Ng et de $3 millions par rapport au coût des *loadmasters*. Des problèmes dérivés de celui de Ng ainsi que des problèmes aléatoires, ont été résolus à l'optimalité. Des études sur le nombre d'items transportés et la variation de la fenêtre de faisabilité du centre de gravité ont été aussi réalisés. Les résultats obtenus nous encouragent à étudier différentes perspectives pour améliorer le temps CPU et pour réduire la mémoire requise pour la résolution de chaque instance.

Nous avons montré par la suite qu'il a été possible d'agréger certaines contraintes liant les items de mêmes caractéristiques. Le modèle obtenu est résolu par la méthode de génération de colonnes imbriquée dans un algorithme de *B&B* en utilisant le critère de branchement de Vanderbeck (2000). A partir des mêmes séries de tests, les résultats démontrent des gains très nets sur les temps de résolution et la mémoire requise de résolution pour chaque instance. Entre autres, le problème réel de Ng est résolu en moins de 2 secondes. Bien que les résultats sont excellents, ils ne sont pas garantis lorsque l'algorithme de plus court chemin avec contraintes sur les ressources est utilisé. D'autres difficultés rencontrées dans cette approche sont la persistance de la symétrie entre les items de même type et la symétrie sur les colonnes générées. Une autre méthode de branchement est proposée par la suite. La règle permet d'éliminer des problèmes de symétrie et d'améliorer en même temps les décisions de branchement. Les résultats des mêmes tests montrent que l'algorithme résout à l'optimalité de façon satisfaisante chaque instance. A titre de comparaison, le problème réel de Ng est résolu en moins de 9 secondes.

Pour le cas général, nous avons proposé deux modèles qui sont des extensions des modèles présentés auparavant. Dans le premier, les items sont chargés selon l'ordre de visite de leur ville de collecte et aucun arrangement de position n'est possible entre les items chargés dans les villes intermédiaires. Une variable de ressource est

ajoutée dans les précédents modèles pour tenir compte des distances parcourues et des villes déjà visitées. Dans le second modèle, des réarrangements de position entre les items déjà chargés dans les villes visitées sont permis. Ce modèle est moins restrictif que le premier. Une procédure d'interdiction de plans est nécessaire pour générer les chargements réalisables. Les deux modèles sont résolus par la méthode de génération de colonnes imbriquée dans un algorithme de *B&B* en utilisant les critères de branchement cités plus haut. Les deux modèles sont testés sur trois cas particuliers de problèmes de chargement. Les résultats des tests montrent qu'une solution est obtenue dans un temps raisonnable pour chaque instance considérée. Au moins un avion est épargné, si on utilise le second modèle. Et pour trouver une "bonne solution" entière avec un saut d'intégrité relativement petit à chaque problème, nous avons proposé une heuristique assez simple.

Les résultats obtenus dans cette thèse démontrent que les modèles proposés se résolvent assez bien avec la méthode de génération de colonnes. A notre connaissance, nous sommes les premiers à résoudre de façon exacte des problèmes de chargements d'avions-cargos entre deux villes qui tiennent en compte les contraintes opérationnelles de chargement et le balancement de l'ensemble avion et cargo. Enfin, nous pensons que les travaux développés dans cette thèse contribuent à l'avancement des méthodes de résolution de problèmes réels similaires et complexes rencontrés en industrie ou dans la vie active.

BIBLIOGRAPHIE

AHUJA R., MAGNANTI T. ET ORLIN J. (1993). Network Flows : Theory, Algorithms and Applications. Prentice Hall, Englewood Cliffs, NJ.

AMIOUNY SAMIR V., BARTHOLDI III J. J., VANDE VATE J. H. ET ZHANG J. (1992). Balanced Loading. *Operations Research* 40, No 2, 238-246.

ANDERSON D. ET ORTIZ C. (1987). AALPS : A Knowledge-Based System for Aircraft Loading. *IEEE Expert*, Winter 1987, 71-79.

ARUNAPURAM S., MATHUR K. ET SOLOW D. (1997). Vehicle Routing and Scheduling with Full Truck Loads. Technical report, Departement of Operations Research and Operations Management, Case Western Reserve University, Ohio.

BARNHART C., JOHNSON E.L., NEMHAUSER G.L., SAVELSBERGH M.W.P. ET VANCE P.H. (1998). Branch-and-Price : Column Generation for Solving Huge Integer Programs. *Operations Research* 46, 316-329.

BISCHOFF E.E ET RATCLIFF M.S.W (1995). Issues in the Development of Approaches to Container Loading. *Omega* 23, 377-390.

BRAMEL J. ET SIMCHI-LEVI D. (1993). On the effectiveness of Set Partitioning Formulations for the Vehicle Routing Problem. Graduate School of Business, Columbia University, NY, 10027.

BROSH I. (1981). Optimal Cargo Allocation on Board a Plane : a Sequential Linear Programming Approach. *European Journal of Operations Research* 8, 40-46.

CHAPEL C.E. (1948). Aircraft Weight, Balance and Loading. *Aero Publishers, Inc.*

COCHARD D. ET YOST K. (1985). Improving Utilization of Air Force Cargo Aircraft. *Interfaces* 15, 53-68.

DANTZIG G.B. ET WOLFE P. (1960). Decomposition Principle for Linear Programs. *Operations Research* 8, 101-111.

DE CARVALHO J.M.V. (1998). Exact Solution of Bin-packing Problems using Column Generation and Branch-and-Bound. *International Transactions in Operational Research* 5, No 1, 35-44.

DEGRAEVE Z. ET SCHRAGE L. (1999). Optimal Integer Solutions to Industrial Cutting Stock Problems. *Informs Journal on Computing*, 11, No 4, 406-419.

DESAULNIERS G., DESROSIERS J., IOACHIM I., SOLOMON M.M. , SOUMIS F. ET VILLENEUVE D. (1998). A Unified Framework for Deterministic Time Constrained Vehicule Routing and Crew Scheduling Problems. T.G. Crainic, G. Laporte, eds. *Fleet Management and Logistics. Kluwer, Norwell, MA*, 57-93.

DESROCHERS M., DESROSIERS J. ET SOLOMON M. (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research*, 40, No 2, 342-354.

DESROCHERS M. ET SOUMIS F. (1988). A Generalized Permanent Labeling Algorithm for the Shortest Path Problem with Time Window. *INFOR* 26, No 3, 191-193.

DESROCHERS M. ET SOUMIS F. (1989). A Column Generation Approach to the Urban Transit Crew Scheduling Problem. *Transportation Science*, 23, 1-13.

DESROSIERS J., DUMAS Y., SOLOMON M.M. ET SOUMIS F.(1995). Time Constrained Routing and Scheduling. *Handbooks in OR & MS, Elsevier Science B.V. Vol. 8 : Network Routing*, M.O. Ball *et al.* (editeurs), North Holland, Amsterdam, 35-139.

DESROSIERS J., SOUMIS F. ET DESROCHERS M. (1984). Routing with time windows by Column Generation. *Networks*, 14, 545-565.

DROR M. (1994). Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW. *Operations Research*, 42, 977-978.

DROR M. ET TRUDEAU P. (1990). Split Delivery Routing. *Naval Research Logistics*, 37, 383-402.

DUMAS Y., DESROSIERS J. ET SOUMIS F. (1991). The Pickup and Delivery Problem with Time Windows. *European Journal of Operational Research*, 54, 7-22.

DYCKHOFF H. (1990). A Typology of Cutting and Packing Problems. *European Journal of Operations Research* 44, 145-159.

DYCKHOFF H. ET FINKE U. (1992). Cutting and Packing in Production and Distribution. Physica-Verlag Heidelberg.

DYCKHOFF H., SCHEITHAUER G. ET TERNO J. (1996). Cutting and Packing : An Annotated Bibliography. Technische Universität Dresden, MATH-NM-08-1996.

EILON S. ET CHRISTOFIDES N. (1971). The Loading Problem. *Management Science*, 17, No 5, 259-268.

GAREY M.R. ET JOHNSON D.S. (1979). Computer and Intractability : a Guide to the Theory of $\mathcal{NP}$-completeness. Freeman, San Francisco.

GEHRING H., MENSHNER K. ET MEYER M. (1990). A computer-based Heuristic for Packing Pooled Shipment Containers. *European Journal of Operations Research* 44, 277-288.

GEOFFRION A.M. (1974). Lagrangean Relaxation for Integer Programming. *Mathematical Programming Study 2*, 82-114.

GILMORE P.C. ET GOMORY R.E. (1961). A Linear Programming Approach to the Cutting Stock Problem. *Operations Research*, 9, No 6, 849-859.

GUEGUEN C. (1999). Méthodes de résolution exacte pour les problèmes de tournées de véhicules. Thèse de Doctorat, Ecole Centrale Paris, France.

HAASE K., DESAULNIERS G. ET DESROSIERS J. (2001). Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems. *Transportation Science*, 35, No 3, 286-303.

HEIDELBERG K., PARNELL G. ET AMES IV J. (1998). Automated Air Load Planning. *Naval Research Logistics*, 45, 751-768.

HUEBNER W.F. (1982). Load Planning, Rapid Mobilization and the Computer. *Air Force Journal of Logistics*, 6, No 1, Winter 1982, 22-24.

IGNIZIO J.P. (1991). Introduction to Expert Systems : The Development and Implementation of Rule-Based Expert Systems. McGraw-Hill, Houston, Texas.

JOHNSON E.L. (1989). Modeling and Strong Linear Programs for Mixed Integer Programming. S.W. Wallace (ed.). *Algorithms and Model Formulations in Mathematical Programming*, NATO ASI Series 51, 1-41.

KOHL N. (1995). Exact Methods for Time Constrained Routing and Related Scheduling Problems. Thèse de Doctorat, Institut of Mathematical Modelling. Technical University of Denmark.

KOHL N. ET MADSEN O.B.G (1997). An Optimization Algorithm for the Vehicle Routing Problem with Time Windows based on Lagrangian Relaxation. *Operations Research* 45, No 3, 395-406.

LARSEN O. ET MIKKELSEN G. (1980). An Interactive System for the Loading of Cargo Aircraft. *European Journal of Operations Research* 4, 367-373.

LEMARÉCHAL C. (1989). Nondifferential Optimization. G.L. Nemhauser, A.H.L. Rinnooy Kan, M.J. Tood (eds.). *Handbooks in Operations Research and Management Science, Volume1 : Optimization.* Elsevier, Amsterdam, 529-572

MARCOTTE O. (1985). The Cutting Stock Problem an Integer Rounding. *Mathematical Programming*, 33, 82-92.

MARTIN-VEGA L.A. (1985). Aircraft Load Planning and the Computer : Description and Review. *Comput. & Indus. Engng*, 9, No 4, 357-369.

MATHUR K. (1998). An Integer-programming-based Heuristic for the Balanced Loading Problem. *Operations Research Letters*, 22, 19-25.

MONGEAU M. ET BES, C. (2000). Optimization of Aircraft Container Loading MIP, Rapport Interne 00-47. Université Paul Sabatier, 31062 Toulouse cedex 04, France.

NEMHAUSER G.L. ET WOLSEY R.A. (1988). Integer and Combinatorial Optimization. Wiley-Interscience, New-york, USA.

NG K.Y.K. (1992). A Multicriteria Optimization Approach to Aircraft Loading. *Operations Research*, 40, No 6, 1200-1205.

RICHARDSON J. (1993). An Expert System for Military Aircraft Load Planning. M.Sc. Thesis, Collège Militaire Royal de St-Jean, Canada.

RYAN D.M. ET FOSTER B.A (1981). An Integer Programming Approach to Scheduling. *Computer Scheduling of Public Transport : Urban Passenger Vehicle and Crew Scheduling*, A. Wren (ed.), North-Holland, Amsterdam, 269-280.

SCHEITHAUER G. ET TERNO J. (1995). A Branch-and-Bound Algorithm for Solving One Dimensional Cutting Stock Problems Exactly. *Applicationes Mathematicae*, 23, No 2, 151-167.

SIERKSMA G. ET TIJSSEN G.A. (1998). Routing helicopters for crew exchanges on off-shore locations. *Annals of Operations Research*, 76, 261-286.

SWEENEY P.E. ET PATERNOSTER E.R. (1992). Cutting and Packing Problems : A Categorized Application-Oriented Research Bibliography. *Journal of Operations Research Society*, 43, No 7, 691-706.

THOMAS C., CAMPBELL K., HINES G. ET RACER M. (1998). Airbus Packing at Federal Express. *Interfaces*, 28, 21-30.

VANCE P. (1996). Branch-and-Price Algorithms for the One-Dimensional Cutting Stock Problem. Working paper. Department of Industrial Engineering, Auburn University, Auburn, Alabama 36849-5346.

VANCE P., BARNHART C., JOHNSON E.L., ET NEMHAUSER G.L. (1994). Solving Binary Cutting Stock Problems by Column Generation and Branch-and-Bound. *Computational Optimization and Applications*, 3, 111-130.

VANDERBECK F. (1994). Decomposition and Column Generation for Integer Programs. Thèse de Doctorat, Université Catholique de Louvain, Belgique.

VANDERBECK F. (1999). Computational Study of a Column Generation Algorithm for Bin Packing and Cutting Stock Problems. *Mathematical Programming* , Vol. 48, No 1, 111-128.

VANDERBECK F. (2000). On Dantzig-Wolfe Decomposition in Integer Programming and Ways to Perform Branching in a Branch-and-price Algorithm. *Operations Research* , Ser. A., 86, 565-594.

VANDERBECK F. ET WOLSEY L.A. (1996). An Exact Algorithm for IP Column Generation. *Operations Research Letters* , 19, No 4, 151-159.

VILLENEUVE D. ET DESAULNIERS G. (2002). The Shortest Path Problem with Forbidden Paths. Les Cahiers du GERAD G-2002-41, Ecole Polytechnique, Montréal.

ANNEXE A : LA DÉCOMPOSITION DE DANTZIG ET WOLFE (1960)

Avec les mêmes notations, le modèle proposé pour ALP se présente sous la forme d'un programme mathématique non linéaire :

$$z_{IP} = \min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij}^k X_{ij}^k \tag{A.1}$$

sujet aux contraintes :

$$\sum_{k \in K} \sum_{(i,j) \in A^k} a_{w,ij}^k X_{ij}^k = q_w, \forall w \in N \tag{A.2}$$

$$\sum_{j:(O^k,j) \in A^k} X_{O^k j}^k = \sum_{j:(j,D^k) \in A^k} X_{jD^k}^k = 1, \forall k \in K \tag{A.3}$$

$$\sum_{j:(i,j) \in A^k} X_{ij}^k - \sum_{j:(j,i) \in A^k} X_{ji}^k = 0, \forall k \in K, \forall i \in V^k \tag{A.4}$$

$$X_{ij}^k \in \{0,1\}, \forall k \in K, \forall (i,j) \in A^k \tag{A.5}$$

$$X_{ij}^k (f_{ij}^{kr}(\mathbf{T}_i^k) - T_j^{kr}) = 0, \forall k \in K, \forall r \in R^k, \forall (i,j) \in A^k \tag{A.6}$$

$$a_i^{kr} \leq T_i^{kr} \leq b_i^{kr}, \forall k \in K, \forall r \in R^k, \forall i \in \{O^k, D^k\} \tag{A.7}$$

$$a_i^{kr} \Big(\sum_{j:(i,j) \in A^k} X_{ij}^k \Big) \leq T_i^{kr} \leq b_i^{kr} \Big(\sum_{j:(i,j) \in A^k} X_{ij}^k \Big), \forall k \in K, \forall r \in R^k, \forall i \in V^k \tag{A.8}$$

Les détails de la procédure de décomposition, tirés de Desaulniers *et al.* (1998), sont décrits dans les sections suivantes.

A.1 Les points extrêmes du sous-problème

La formulation proposée présente des contraintes liantes qui seront retenues par le problème maître et une structure diagonale formant $|K|$ sous-problèmes indépendants. Le principe de la décomposition consiste à décrire l'ensemble des solutions de chaque sous-problème au moyen d'une combinaison convexe de ses points extrêmes. Pour chaque $k \in K$ est associé un sous-problème qui est constitué des contraintes (A.3)- (A.8), et d'une fonction objectif que nous définirons plus tard. Comme les contraintes d'un sous-problème k définissent une structure de chemin sur laquelle une unité de flot est envoyée entre O^k et D^k, ses points extrêmes correspondent à des chemins dans G^k ou dans notre cas, à des patrons de chargement réalisables pour un type de *chalk* k, et sont décrits par les vecteurs de flots et ressources :

$$(\mathbf{x}_e^k, \tau_e^k) = (x_{ije}^k, \tau_{ie}^{kr}), k \in K, e \in \Omega^k, (i,j) \in A^k, r \in R^k$$

où Ω^k, indexé par e, dénote l'ensemble de ces points extrêmes. Chaque vecteur $(\mathbf{x}_e^k, \tau_e^k)$ est une solution admissible vérifiant les contraintes d'un sous-problème k. Ainsi, toutes les solutions X_{ij}^k et T_i^{kr} satisfaisant aux contraintes (A.3)- (A.8) peuvent être exprimées comme une combinaison convexe non négative des patrons de chargement générés par le sous-problème au moyen des équations suivantes :

$$X_{ij}^k = \sum_{e \in \Omega^k} x_{ije}^k \theta_e^k, \forall k \in K, \forall (i,j) \in A^k \tag{A.9}$$

$$X_{ij}^k \in \{0,1\}, \forall k \in K, \forall (i,j) \in A^k \tag{A.10}$$

$$T_i^{kr} = \sum_{e \in \Omega^k} \tau_{ie}^{kr} \theta_e^k, \forall k \in K, \forall r \in R^k, \forall i \in V^k \tag{A.11}$$

$$\sum_{e \in \Omega^k} \theta_e^k = 1, \forall k \in K \tag{A.12}$$

$$\theta_e^k \geq 0, \forall k \in K, \forall e \in \Omega^k. \tag{A.13}$$

Les nouvelles variables de décision $\theta_e^k, k \in K, e \in \Omega^k$, sont appelées des variables de plan de chargement. Notons qu'une variable de plan de chargement est associée, entre autres, au plan de chargement à vide.

A.2 Le problème maître en nombres entiers

Le problème maître est constitué par la fonction objectif (A.1) et les contraintes (A.2), et contient de ce fait toutes les contraintes impliquant plus d'un avion. En substituant (A.9)- (A.13) dans (A.1) et (A.2), le problème maître devient :

$$\min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij}^k \left(\sum_{e \in \Omega^k} x_{ije}^k \theta_e^k \right) \tag{A.14}$$

sujet aux contraintes :

$$\sum_{k \in K} \sum_{e \in \Omega^k} \left(\sum_{(i,j) \in A^k} a_{w,ij}^k x_{ije}^k \right) \theta_e^k = q_w, \forall w \in N \tag{A.15}$$

$$(A.9),\ (A.10),\ (A.12) \quad et \quad (A.13).$$

La proposition suivante permet de simplifier les expressions dans le problème maître :

Proposition 6 *Soient θ_e^k et $X_{ij}^k, \forall k \in K, \forall e \in \Omega^k$ et $\forall (i,j) \in A^k$ des variables satisfaisant (A.9), (A.12) et (A.13). Alors toutes les variables de plan de chargement θ_e^k prennent des valeurs binaires si et seulement si toutes les variables de flot X_{ij}^k prennent des valeurs binaires.*

Preuve : Voir Desaulniers *et al.* (1998).

Comme la relation (A.12) et la Proposition 1 ci-dessus impliquent que, pour chaque $k \in K$, une seule variable de plan de chargement prend la valeur 1 tandis que toutes

les autres sont nulles alors :

$$c_{ij}^k(\sum_{e\in\Omega^k} x_{ije}^k\theta_e^k) = \sum_{e\in\Omega^k} c_{ij}^k x_{ije}^k\theta_e^k, \qquad \forall k \in K.$$

Si nous définissons les nouveaux coefficients c_e^k et a_{we}^k comme suit :

$$c_e^k = \sum_{(i,j)\in A^k} c_{ij}^k x_{ije}^k, \forall k \in K, \forall e \in \Omega^k \tag{A.16}$$

$$a_{we}^k = \sum_{(i,j)\in A^k} a_{w,ij}^k x_{ije}^k, \forall k \in K, \forall w \in N^k, \forall e \in \Omega^k \tag{A.17}$$

en les substituant dans (A.14) et (A.15) et en arrangeant l'ordre de la sommation, le problème maître s'écrit :

$$\min \sum_{k\in K}\sum_{e\in\Omega^k} c_e^k\theta_e^k \tag{A.18}$$

sujet aux contraintes :

$$\sum_{k\in K}\sum_{e\in\Omega^k} a_{we}^k\theta_e^k = q_w, \forall w \in N \tag{A.19}$$

$$\sum_{e\in\Omega^k} \theta_e^k = 1, \forall k \in K \tag{A.20}$$

$$\theta_e^k \in \{0,1\}, \forall k \in K, \quad \forall e \in \Omega^k. \tag{A.21}$$

Le premier ensemble de contraintes (A.19) exige que, pour chaque type d'items $w, w \in N$, le transport soit effectué par des chargements appropriés. En effet, $a_{we}^k, w \in N^k, e \in \Omega^k, k \in K$ est une constante positive et elle indique le nombre de fois qu'un exemplaire de l'item w est chargé dans l'avion k pour le plan de chargement e. Les contraintes de convexité (A.20) pour chaque avion $k \in K$ assure qu'un seul plan de chargement est assigné à cet avion k. Ces contraintes correspondent à (A.3), dans la formulation originale, c'est-à-dire,

$$\sum_{j:(O^kj)\in A^k} X_{O^kj}^k = \sum_{j:(jD^k)\in A^k} X_{jD^k}^k = 1.$$

La formulation (A.18) - (A.20) et la non-négativité est la relaxation linéaire d'un problème de type partitionnement avec des contraintes additionnelles de convexité, elle est utilisée pour évaluer une borne inférieure sur la solution optimale entière du modèle multiflot.

A.3 La fonction objectif du sous-problème

Soient $\pi = (\pi_w \quad | \quad w \in N)$ et $\gamma = (\gamma^k \quad | \quad k \in K)$ les vecteurs des variables duales associées respectivement aux contraintes (A.19) et (A.20). Le coût réduit $\bar{c}_e^k(\pi, \gamma)$ d'une variable de plan de chargement $\theta_e^k, k \in K, e \in \Omega^k$, est donné par :

$$\begin{aligned} \bar{c}_e^k(\pi,\gamma) &= c_e^k - \sum_{w \in N^k} a_{we}^k \pi_w - \gamma^k \\ &= c_e^k - \sum_{(i,j) \in A^k} \Big(\sum_{w \in N^k} a_{w,ij}^k \pi_w \Big) x_{ije}^k - \gamma^k. \end{aligned}$$

Pour trouver la variable de plan de chargement ayant le moindre coût réduit parmi les $\theta_e^k, e \in \Omega^k$, la fonction objectif du sous-problème est :

$$\min \quad c_e^k - \sum_{(i,j) \in A^k} \Big(\sum_{w \in N^k} a_{w,ij}^k \pi_w \Big) X_{ij}^k - \gamma^k. \tag{A.22}$$

Pour chaque $k \in K$, le sous-problème est constitué de cette fonction objectif (A.22) et des contraintes (A.3) à (A.8). Ce problème d'optimisation correspond à un problème de plus court chemin avec contraintes de ressource.

A.4 Agrégation des commodités

Nous pouvons simplifier la formulation (A.18) à (A.20). Remarquons que tous les avions-cargos sont identiques, les sous-problèmes deviennent alors équivalents et admettent le même ensemble de solutions admissibles Ω. Posons : $c_e = c_e^k$ et $a_{we} = a_{we}^k, k \in K, w \in N, e \in \Omega$. c_e représente le coût du plan de chargement e et a_{we} représente la contribution du plan de chargement e pour assurer le transport des exemplaires de l'item w. De même, si nous définissons les variables $\theta_e = \sum_{k \in K} \theta_e^k, e \in \Omega$, θ_e donne le nombre d'avions utilisant le plan de chargement e. Comme les variables θ_e^k sont positives, les variables θ_e doivent l'être aussi. D'après (A.20), nous avons :

$$\sum_{k \in K} \sum_{e \in \Omega^k} \theta_e^k = \sum_{k \in K} 1 \Longleftrightarrow \sum_{e \in \Omega} \sum_{k \in K} \theta_e^k = |K| \Longleftrightarrow \sum_{e \in \Omega} \theta_e = |K|. \tag{A.23}$$

La relaxation linéaire du problème maître (A.18) à (A.20) est équivalente à :

$$\min \sum_{e \in \Omega} c_e \theta_e \tag{A.24}$$

sujet aux contraintes :

$$\sum_{e \in \Omega} a_{we} \theta_e = q_w, \forall w \in N \tag{A.25}$$

$$\theta_e \geq 0, \forall e \in \Omega \tag{A.26}$$

$$\theta_e = \sum_{k \in K} \theta_e^k, \forall e \in \Omega \tag{A.27}$$

$$\sum_{e \in \Omega} \theta_e^k = 1, \forall k \in K \tag{A.28}$$

$$\theta_e^k \geq 0, \forall k \in K, \forall e \in \Omega. \tag{A.29}$$

Les résultats suivants nous permettent de tirer profit de cette formulation agrégée. A chaque solution fractionnaire θ_e de (A.24) à (A.26) est associée une autre en

termes de variable désagrégée θ_e^k qui répond aux contraintes (A.27)- (A.29). En effet, supposons que $\theta_e, e \in \Omega$, est une solution et pour chaque $\theta_e^k, k \in K, e \in \Omega$, posons :

$$\theta_e^k = \frac{\theta_e}{|K|}, \forall k \in K, \forall e \in \Omega.$$

D'après (A.23), nous pouvons vérifier que cette solution vérifie les contraintes (A.27)-(A.29). Ainsi, toute solution à partir des variables agrégées $\theta_e, e \in \Omega$ peut être convertie en solution sous forme des variables désagrégées $\theta_e^k, k \in K, e \in \Omega$, de sorte que le problème (A.24) à (A.26) peut être utilisé comme le problème maître du modèle. Par ailleurs, si nous considérons une solution θ_e^* entière de (A.24)- (A.26), nous pouvons convertir cette dernière en termes de variables θ_e^{*k} en assignant chaque plan de chargement non vide à un ou plusieurs avions non utilisés, selon le besoin. Cela est toujours possible car d'après (A.23) le nombre d'avions chargés est au plus $|K|$. Les avions non chargés dans la solution optimale sont affectés par le plan de chargement vide et cette solution vérifie bien (A.27)- (A.29).

Notre approche va donc se limiter à cette dernière formulation agrégée (A.24)-(A.26) pour le problème maître qui contient $|K|$ fois moins de variables et a moins de contraintes que la formulation (A.18)- (A.20).

ANNEXE B : LA GÉNÉRATION DES DONNÉES

B.1 La génération des quantités des items

Les quantités pour chaque type d'item sont générées selon une loi uniforme discrète entre 1 et 50.

B.2 La génération des longueurs des items

Les longueurs sont générées selon une loi de probabilité trapèzoïdale. En effet, nous avons approché la densité sous la forme d'un trapèze (Figure B.1) dont les coins sont donnés par les coordonnées suivantes : $(a, 0)$, (b, y), (c, y), $(d, 0)$. La valeur y est déterminée pour avoir une surface de 1 unité et les valeurs de a, b, c et d sont déterminées pour avoir une distribution similaire aux données de Ng (1992). Ainsi, pour chaque type d'item, on génére un nombre aléatoire r selon une loi uniforme $(0, 1)$. A partir de cette valeur, nous avons la *longueur* générée suivante :

$$longueur = \begin{cases} \sqrt{15000r} + 50 & \text{si } 0 < r \leq 0,166, \\ 150r + 75 & \text{si } 0,166 < r \leq 0,5, \\ -\sqrt{45000(1-r)} + 300 & \text{si } 0,5 < r \leq 1. \end{cases} \tag{B.1}$$

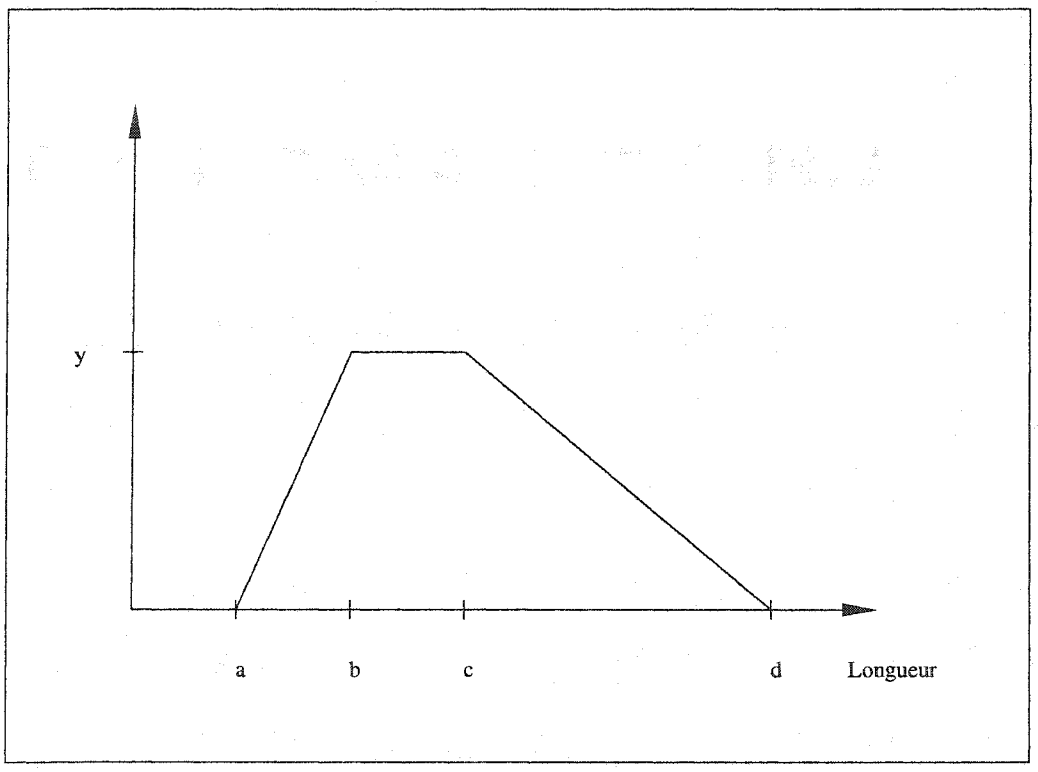

Figure B.1 – Loi de probabilité utilisée pour les longueurs

B.3 La génération des poids des items

Les poids sont générés selon une loi de probabilité décrit à la figure B.2. Les coins sont donnés par les coordonnées suivantes : (a, y_2), (b, y_2), (b, y_1), $(c, 0)$. Les valeurs y_1 et y_2 sont déterminées pour avoir une surface de 1 unité et les valeurs de a, b et c sont déterminées pour avoir une distribution similaire aux données de Ng (1992). Ainsi, pour chaque type d'item, on génére un nombre aléatoire r selon une loi uniforme $(0, 1)$. A partir de cette valeur, nous avons le *poids* généré suivant :

$$poids = \begin{cases} 10 + \frac{1000r}{8} & \text{si } 0 < r \leq 0,72, \\ 350 - 10000\sqrt{\frac{1-r}{448}} & \text{si } 0,72 < r \leq 1. \end{cases} \tag{B.2}$$

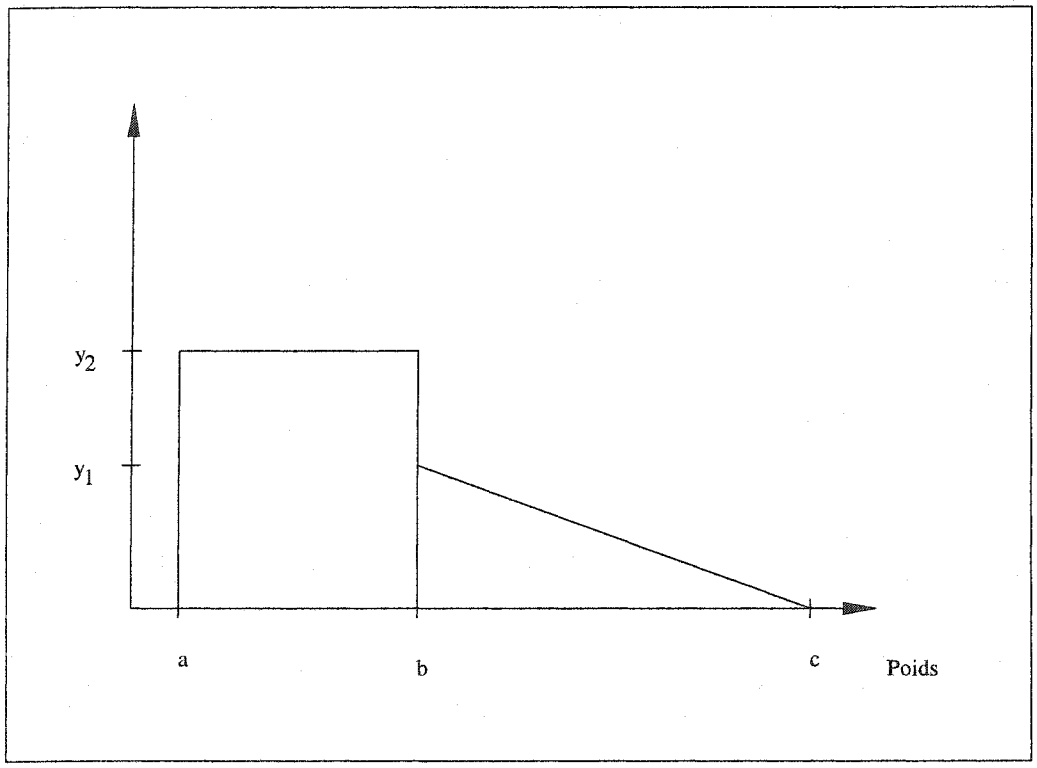

Figure B.2 – Loi de probabilité utilisée pour les poids